A Framework to Aggregate and Report Textual Data Without Editorial Bias

By Keerat Sharma

Submitted to the Department of Computer Science
in Partial Fulfillment
of the Requirement for the Degree of

Master of Science
In

Computer Science

At

Hofstra University

2011

Professor Krishnan Pillaipakkamnatt, Advisor

Professor Habib Ammari, Department of Computer Science

Professor Xiang Fu, Department of Computer Science

A Framework to Aggregate and Report Textual Data Without Editorial Bias

Master's Thesis in Computer Science
Hofstra University

Keerat Sharma

June 3, 2011

Abstract

A system to harvest facts and their relationships while eliminating editorial bias from the news in near-real-time is presented. In addition, the implementation of wordsinmedia.com is documented as a working implementation of the described framework.

Acknowledgements

This body of work came to fruition because Dr Krishnan Pillaipakkamnatt ("Dr Krish") pushed me to spend an extra semester to take it from a Project to a Thesis. This turned out to be a challenging and rewarding experience, and I'm glad that Dr Krish was my Advisor and mentor through this process.

I must thank Dr Habib Ammari, who was a constant source of positivity and encouragement during his lectures and with my thesis as a Co-Advisor.

Dr Xiang Fu has been an advisor throughout my graduate education. In his capacity as the Chairman of the Distance Learning Program for Computer Science, he helped with many details of the program at any hour the day. I appreciate the numerous opportunities and guidance he has provided me.

Numerous friends have provided me with ideas, editorial support, and debate around my thesis who I am grateful for, including Pohl Longsine, David Milligan, Michael Arabia, Scott Ogden, Kristine Ogden, Vishal Santoshi, Sethu Kalavarkur, and Imran Khan.

Finally, I must acknowledge the support I got from my parents and wife. In particular, my wife has had to deal with my thesis induced nocturnal working habits while providing me with advice and input around the website and document that improved both significantly. This would not have been possible without her help and support.

Contents

Chapter 1

Introduction

News aggregation systems[1] have typically focussed on clustering similar articles from different media outlets together[2]. All of these systems preserve and present originally published content. We contend that this is problematic as it maintains the publisher's editorial bias and their specific viewpoint on a topic. We propose a system that decomposes news in near real time, and automatically identifies themes and concepts within and across articles. This system allows a reader to get an aggregated perspective on the news, from multiple sources, without having to read through published content from any media outlet.

To illustrate the problem further, consider that after the 2011 State of the Union address was delivered by President Obama, it took only five hours for more than 9,000 fresh articles from news agencies across the world to circulate

[1]Examples include Google News, Yahoo! News, and bing News
[2]See Appendix A

across the internet. Given the magnitude of the volume of content available, a reader frequently has no choice but to accept that the subset of articles that they happened to read were truthful, and that they comprehensively covered the topic of interest. We find this to be inefficient and unrealistic given the volume of data in consideration.

Fox News

Obama lauds education, technology in State of the Union address
USA Today - Mimi Hall - **41 minutes ago**
"At stake is whether new jobs and industries will take root in this country or somewhere else," Obama said in his second State of the Union Address.
Video: Obama: Leadership Now 'Shared Responsibility' The Associated Press
State of the Union: Obama calls for unity, yet challenges GOP Los Angeles Times
Huffington Post - CBS News - New York Daily News - The Guardian
all 9,214 news articles »

Figure 1.1: President Obama's State of the Union Address on January 25th had generated over 9,000 articles that Google News had already indexed by 9 PM PST.

Our solution applies concepts that exist today to aggregate structured data to get a comprehensive perspective. Consider a busy web server: it may get millions, or billions of requests a day. Each of these requests are logged as individual entries. It is generally accepted that each entry can not be scanned by an individual that wishes to understand the health or traffic patterns of the web server. Instead, reporting tools take over by aggregating each log entry into buckets based in time, or other vectors. These tools deliver comprehensive coverage by ensuring that each individual log line is taken into account by generating an aggregate or some other metric. This strategy eliminates the bias from a single log statement by ensuring that it is seen in an aggregate context. We propose to apply such techniques to the news.

News is unstructured textual data. Structured data like logs from a web server, are easy to analyze due to their inherent structure. Reporting and analysis tools for structured data are frequently available, or easy to build. However, it is possible to take inherently unstructured textual data, like the news, and decompose it into a formal data structure at a granular level that can be subjected to the same types of analysis that is common with structured data. The news can indeed be analyzed and aggregated in a similar fashion as a financials spreadsheet.

This thesis documents the construction of wordsinmedia.com, a system that provides aggregated news with thematic linkages between terms. Chapter 2 focusses on article acquisition over the internet and the subsequent processing needed to make content ready for processing. Chapter 3 examines decomposing article content into sentences and identifying relevant terms to build a data structure that represents an article. Chapter 4 describes a database schema and iterates through a selection of reporting possibilities. Chapter 5 walks through some lessons learned and future research avenues.

As noted earlier, the described work is implemented at wordsinmedia.com, which serves as a prototype, and reference around the reporting and analysis capabilities.

Chapter 2

Acquiring News

To create a system that handles the analysis of news data, a reliable mechanism to acquire content must be present. Fortunately, almost all news outlets provide their content via Really Simple Syndication (RSS)[4] feeds on the internet. Unfortunately, the articles in these feeds are meant to be read by people, not parsed by machines. Thus, a variety of issues must be overcome before RSS based news content can actually be parsed by an analysis component.

For the purposes of this research, the scope was limited to parsing English news only. The base unit of meaning in English is a word. Many words sequenced together form a sentence. Typically, the context of a word can only be inferred when you analyze the sentence it occurs within. While there are numerous ways to parse English, none of them are perfect and the eccentricities of the language pose numerous challenges. However, by

combining strategies and by operating within certain limits, it is possible to parse English reliably and to have a sentence level data structure that decomposes to words and their context within the sentence.

Armed with sentence level analysis, it is possible to apply this iteratively within an article to construct an article level data structure. This consists of many parts including nouns, various verbs, their corresponding frequencies and more. The data structure is easily normalized, mutated, and further analyzable across the scope of all articles analyzed to date. The more vectors preserved (time, source of article, nouns present) the more the possibilities for analysis.

The rest of this chapter explores each of the above phases in depth.

2.1 An Overview of RSS

RSS is a protocol to share content that is updated periodically. RSS is generally implemented using HTTP[1] as a transport layer. A request to an RSS URL (frequently referred to as a "feed") results in an XML document response (Figure 2.1). The XML document can conform to one of a number of variants and versions within the RSS family (ATOM[2], RDF (RSS1)[3], RSS2[4], etc). Most RSS parsing libraries are equipped to transparently manage the variants and provide a uniform data structure to interact with after parsing the RSS XML document[6][7].

Each RSS feed can contain multiple channels (eg: headlines, sports, fash-

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:blogChannel="http://a.b.com/rss">
<channel>
 <title>Example RSS</title>
 <link>http://a.b.com/</link>
 <description>An example ..</description>
 <language>en-us</language>
 <lastBuildDate>
   Mon, 30 Sep 2002 11:00:00 GMT
 </lastBuildDate>
 <ttl>40</ttl>
 <item>
   <title>Example entry</title>
   <description>
     Typically, some content of interest, like an article
   </description>
   <pubDate>Mon, 30 Sep 2002 01:56:02 GMT</pubDate>
   <link>http://a.b.com/story_url</link>
   <guid>unique string</guid>
 </item>
</channel>
</rss>
```

Figure 2.1: A sample RSS document with just one item

ion), but it is common for a feed to have just one *channel*. Each *channel* has meta-data to describe itself, like its *title*, *description* and *language*. Within each *channel* there is a list of items. An *item* is a unit of content. A *channel* provides a *lastBuildDate* field which states when the *channel* was last updated with content. This is what most RSS clients store and compare with new RSS feed fetches to see if the *channel* should be parsed for fresh items.

Most RSS clients poll an RSS feed with a certain frequency, typically about fifteen minutes. Some take the channel's *ttl*[1] field into account, which provides the client with the feed publisher's requested poll frequency for subscribers.

For news outlets, each *item* within a *channel* is a story. Items are struc-

[1]Time to live

tured minimally, with just a *title*, *description*, publish date (*pubDate*), a *link*, and a globally unique identifier (*guid*). The *title* and *description* provide all the viewable content to a story. The *link* is typically a URL to an online copy of the story. Most news providers truncate the RSS version of a story, so the *link* is important if the reader wishes to read the rest of the story in full. An item's *guid* provides a globally unique identifier for the story. RSS clients are expected to use the *guid* to ascertain if they have already encountered the *item* on a previous document fetch/parse cycle.

2.2 RSS in the Real World

```
my $feed = XML::FeedPP->new($rssFeedURL);
foreach my $item ($feed->get_item()){
  print $item->title();
  print $item->description();
}
```

Figure 2.2: Fetching an RSS feed and iterating through its items using XML::FeedPP [7]

While fetching and parsing RSS items is a relatively trivial operation (Figure 2.2), the contents of both the *title* and *description* have no structural guidelines beyond being XML safe. This leads to a plethora of problems.

Real world RSS is quite polluted from a textual analysis standpoint (Figure 2.3). Primarily, most news outlets use embedded advertising to monetize their content. Additionally, since most RSS feed content is rendered by a client for a user to read, the addition of embedded images, bolded or itali-

```
<title>U.S. Nuclear Industry Faces New Uncertainty</title>
<description>A fragile bipartisan consensus on nuclear power&#x2019;s promise
for the United States may have dissolved.&lt;br clear=&quot;both&quot; style=&quot;clear:
both;&quot;/&gt;&lt;br clear=&quot;both&quot; style=&quot;clear: both;&quot;/&gt;&lt;a
href=&quot;http://ads.pheedo.com/click.phdo?s=0159e1d628b59e2ae52f62785ae17898&amp;p=1&quot;
&gt;&lt;img alt=&quot;&quot; style=&quot;border: 0;&quot; border=&quot;0&quot;
src=&quot;http://ads.pheedo.com/img.phdo?s=0159e1d628b59e2ae52f698
&amp;p=1&quot;/&gt;&lt;/a&gt;
</description>
```

Figure 2.3: An example of real world RSS item's description (taken from NYTimes.com on March 13 2011)

cized text or other embellishments add to the end user reading experience, but create non-content artifacts. Finally, since RSS is itself an XML format, all content within the *title* or *description* fields must fit the standards for $CDATA$[2] in XML. Common symbols in a normal sentence like an apostrophe must be escaped to allow for parsing the RSS enveloping structure successfully. Figure 2.4, shows how the XML content in Figure 2.3 would appear to an end user using an HTML capable RSS client.

Figure 2.4: Rendered version of XML in Figure 2.3

[2]See CDATA Sections in the XML5 spec: 2.7 CDATA Sections

To successfully analyze the *title* and *description* as textual content, they need to be parsed as HTML content. This eliminates the special character encoding problems. The description payload from Figure 2.3 can be seen converted to HTML in Figure 2.5.

```
A fragile bipartisan consensus on nuclear power's promise for the United States may have
dissolved.<br clear="both" style="clear: both;"/><br clear="both" style="clear: both;"/><a
href="http://ads.pheedo.com/click.phdo?s=0159e1d628b59e2ae52f62785ae17898&p=1" ><img alt=""
style="border: 0;" border="0" src="http://ads.pheedo.com/img.phdo?s=0159e1d628b598
&p=1"/></a>
```

Figure 2.5: After HTML conversion

The last phase of parsing involves weeding out HTML decorations like images and links to finally produce analyzable content:

> "A fragile bipartisan consensus on nuclear power's promise for the United States may have dissolved. "

2.3 Systematic processing

The starting point for an RSS client is to have an RSS feed URL at hand. The set of previously parsed items must be recorded to know which items in a freshly polled feed are new. Figure 2.6 illustrates how a proposed system can make use of a durable store to manage RSS content.

Periodically, a module is fired by some timer (cron[3]). The first step is to acquire a set of feed URLs from a durable location. Each feed is iterated over till there are no more feeds to analyze. For each feed, a fresh document is

[3]A configurable time based scheduler. See http://en.wikipedia.org/wiki/Cron

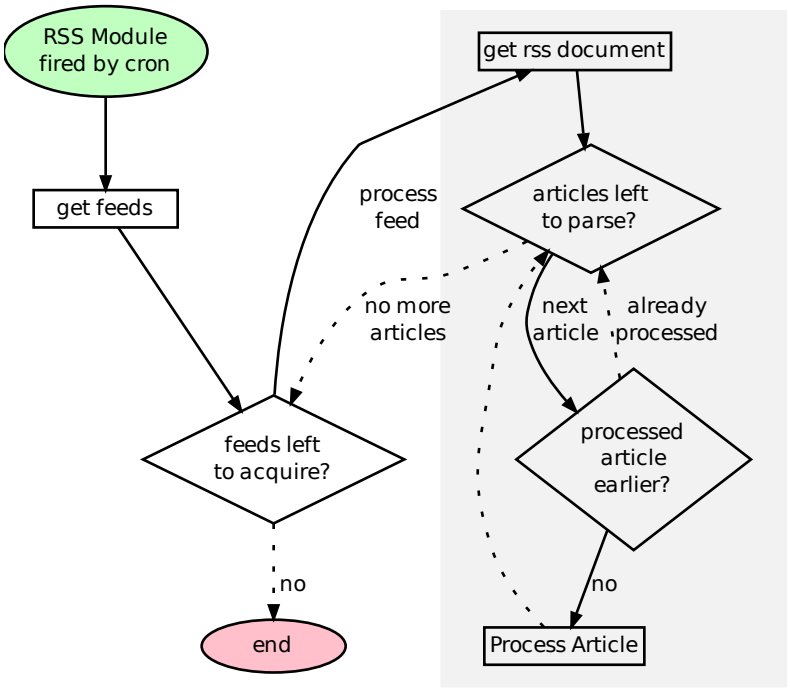

Figure 2.6: Process flow for RSS processing

fetched via HTTP. Each RSS document is parsed and the items are analyzed in a loop. If an *item* (article) has already been parsed, we skip processing it. If not, the *item* forms the basis of what needs to be analyzed. Its *title*, *description*, *pubDate* and other fields are extracted, cleaned and dispatched for processing. This sets up a framework wherein news outlets are registered and polled periodically. Fresh content is harvested and cleaned, and made ready for actual textual analysis.

Chapter 3

Analyzing Text

Text analytics, text mining, semantic analysis and related fields focus on the analysis of textual data. To frame the complexity behind unstructured textual data analysis, consider getting a financial report in two possible formats. The first is descriptive, perhaps from a press statement:

> Our buyer and seller data is limited to just three sellers. Acme bought seventeen, while Bob purchased only twelve. Alice was the lowest purchaser, acquiring only three units.

Figure 3.1: Purchase data in a descriptive paragraph

Getting data in the above format isn't normal for many reasons. It is hard to isolate the core facts from the descriptive text that is interleaved. Additionally, the descriptive format doesn't present an easy way to compare, re-organize, or analyze the data effectively. Consider the scale of this problem if there are hundreds of data points. A natural tendency is to organize and

maintain data in a much more structured fashion. Figure 3.2 shows a tabular representation of the same content presented in Figure 3.1.

Buyer	Units
Acme	17
Bob	12
Alice	3

Figure 3.2: A table of data

The data in the table is inherently structured: each cell has context via its column header and/or row label. This automatic contextualization provides explicit meaning to each cell. Further, since each data point is an atomic value, we can string the data points together, creating graphs or other enrichments to the data like computing an average. This method scales well. As the data grows, there are simple ways to sift through the set.

This chapter introduces the concept of Link Grammar, explains how we use it in our framework to create sentence level data structures, and concludes with the data structure for an article: the Article Histogram.

3.1 Using Link Grammar

One way to identify the structure of a sentence is to use a word based parsing strategy such as Link Grammar[8]. In Link Grammar, a dictionary defines words and associates each word with a set of possible inbound and outbound edges. The polarity of the edges force ordering of words in a sentence, but don't necessarily dictate adjacency. Sentences are considered to be graphs of words, linked by their possible edges in a legal manner.

A formal implementation of a Link Grammar parser with an English dictionary was first produced in 1991 by Daniel Sleator and Davy Temperley [8] at Carnegie Melon University. The parser has grown in sophistication since, and the accompanying dictionaries have expanded to additional languages including Lithuanian and German. It is implemented in ANSI C, but there are many wrappers available in a broad range of other programming languages.[1]

AbiWord[2] began using the parser to provide grammar checking in its word processor. Logistical and licensing issues steered the Link Grammar project to be taken over by AbiWord. Today, Link Grammar is available via the liblinkgrammar package on most operating systems, and its development continues. Numerous other linguistic projects make use of Link Grammar for research or other uses. [3]

3.1.1 Link Grammar at a High Level

Working with examples is an easy way to understand how Link Grammar works from a conceptual standpoint. The Link Parser requires a dictionary, from which it sources a base set of words, and their legal inbound and outbound edges. As an example, consider the simplistic dictionary in Figure 3.3.

Consider the two words: “phone” and “the”. Let’s examine how they can

[1] Delphi, Java, OCaml, Python, Ruby and more
[2] An opensource word processor
[3] Eg: RelEx Semantic Relation Extractor, BioLG, Pre-parsed Wikipedia

Word	Inbound	Outbound
loudly	Adverb	
phone	Adjective, Determiner, Noun Modifier	Verb
rang	Verb	Adverb
yellow	Noun	Adjective
the		Determiner

Figure 3.3: A sample link grammar dictionary

be legally linked:

- "phone" has "Determiner" in the set of legal inbound edges,
- "the" has only one possible edge: an outbound connection of type "Determiner",
- we can therefore link "the" with "phone", with "the" before "phone", via a "Determiner" edge,
- note that it would be illegal for us to link "the" to any other word in our dictionary defined above since none of them accept an inbound "Determiner" edge, except "phone".

Let's consider the following sentence: "The phone rang." We can represent this graphically, as seen in Figure 3.4. As evident, there are two edges that string the three words together and the sentence is conformant with the Link Grammar described in Figure 3.3.

As a contrast, consider the malformed sentence: "The rang." as shown in Figure 3.5. "The"'s sole outbound edge is not a match for any of the possible

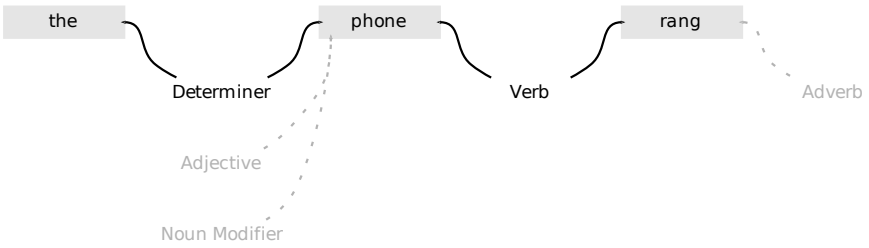

Figure 3.4: Links in "The phone rang."

inbound edges that "rang" can legally accept. As such, "The rang" can not be a legal sentence on its own.

Figure 3.5: Links in "The rang."

There are cases where Link Grammar has to deal with words it does not have in its dictionary. Common examples of this include names of people, places, slang and so on. Let's consider the following sentence: "The Acme phone rang", as seen in Figure 3.6. Note that "Acme" is not a defined entity in our dictionary (Figure 3.3). However, the Noun Modifier is an edge type that has specific rules that allow us to infer that the word preceding "phone", while not located in the dictionary is a Noun Modifier.

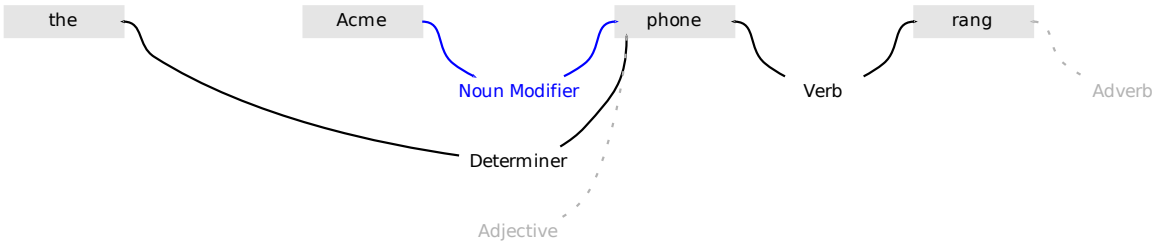

Figure 3.6: Links in "The Acme phone rang."

3.1.2 Real World Cases

Link Grammar ships with a broad dictionary of words and edge types[4]. Edge types are typically one or two upper case alphabets. In some case, the edge is contextualized with a lower case suffix that qualifies the edge further.

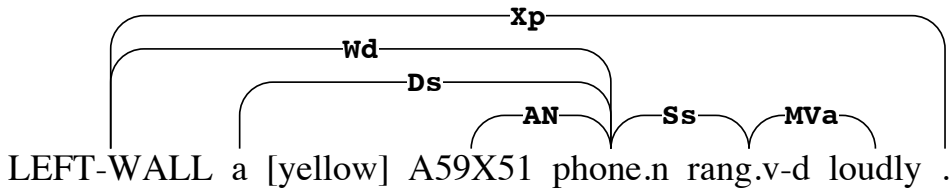

Figure 3.7: Linkages for "A yellow A59X51 phone rang loudly."

In order to understand how thorough Link Grammar is, and how effective the parser and dictionaries are, we will examine Figure 3.7. The sentence fed to the parser is "A yellow A59X51 phone rang loudly". We chose "A59X51" as a random word that we would not find in the supplied dictionary.

Notice that the parser adds a left wall to the sentence structure. This left wall is an explicit marker that indicates the starting point of a sentence. "X is used to connect punctuation symbols to words", with Xp connecting to a

[4]The base English dictionary (including edge types) is about two megabytes

period. Therefore, the left wall's Xp linkage always encapsulates the entire sentence.

Let's examine the Wd connection from the left wall to "phone". Per the Link Grammar reference, "W is used to attach main clauses to the wall". As evident, the sentence is inherently about the phone- the specific noun clause in the sentence that is being described. The "d" suffix is a declarative indicator, establishing that the sentence is a declaration about a specific noun- a phone in this case. The Ds linkage between "a" and "phone" is a determiner edge, with the s suffix denoting a singular binding, reflective of the singular phone in this example.

The AN edge that binds "A59X51" to "phone" is a noun-modifier connection. The word "phone" has been qualified as a noun within the sentence structure and was found to have an available noun modifier edge. Thus, the unrecognizable prefixing word "A59X51" was bound to "phone" as a modifier. This is an especially powerful capability when it comes to parsing the news since names, places and other terms that may not already be registered in a dictionary are able to be contextualized by their surrounding words.

The Ss edge linking "phone" to "rang" is a subject noun linkage to a verb within a singular context. Finally, the MVa linkage connects "rang" to "loudly" with a verb to adverb binding.

Since the linkages are all directional, it is possible for the parser to qualify how a word has been used in the context of the sentence. To this degree, some of the words have suffixes attached to them after parsing. Notably,

"phone.n" is indicative that "phone" has been contextualized as a noun. Additionally, "rang" is a verb with specific ordering via its suffix of "v-d".

3.1.3 Challenges with Link Grammar

While Link Grammar is an extremely effective mechanism to parse sentences, it does have a number of drawbacks. Foremost, Link Grammar's effectiveness is predicated on parseable sentences. Alternatively, sentences can be very complex, creating a high degree of complexity within the parsing phase. This adds a lot of time to the parse, sometimes measured in seconds. As such, an article with approximately fifty sentences could take up to a minute to parse.

With the news, the quality of data is variant, and sometimes sentences aren't formed properly. While Link Grammar can generally tolerate some degree of error rate, there are low percentages that tend to fail altogether. Malformed quotation marks, imbalanced parentheses, and other artifacts within sentences cause the Link Parser to fail sporadically. Some leading sentences of articles in particular cause trouble where capitalization is not regular.

These various constraints force us to accept that some sentences will not be parsed using Link Grammar. As a fallback, sentences are tokenized to words and examined using the WordNet database. This allows us to extract nouns, verbs and adjectives. Note that nouns like people's names and places in the news don't always match in the WordNet database, so this is a last resort.

3.2 Article Data Structures

The purpose of all of the analysis described in the Link Grammar section is to be able to create a data structure that represents an article. This must be a data structure that can be used to compare one article to another, or to find specific elements, or any other deterministic operation. To do so, we work our way from a granular data structure that represents a sentence and aggregate its contents up to an article.

3.2.1 Per-Sentence Data Structures

We analyze an article, sentence by sentence, so that we can create a data structure that reflects the contents. This is the basis of our data structure that represents the unstructured article content. Each examination of a sentence is expected to provide a set of words that are relevant to the sentence. Consider the following sentence from a news article:

> That morning newspapers carried the story of John Galliano's dismissal from Dior.

On parsing the sentence, we get an array of tuples like so:

```
[newspapers:  newspapers.n,
story:  story.n,
John Galliano:  John.m Galliano,
morning:  morning.n-u,
John:  John.m,
dismissal:  dismissal.n-u]
```

Each tuple's first element is the word or phrase as seen in the parsed sentence. The second element in the tuple is the parsed value with Link Parser context suffixed to the word. For WordNet parsed sentences, the latter part is null and meaningless. This data structure harvests words and phrases of interest while eliminating sections of grammar that may not be useful. This mechanism is configurable, such that more grammatical linkages can be pulled in, or fewer. In the above example, the focus is on noun and verb contexts.

3.2.2 Aggregating Sentences

To be able to create a data structure that adequately represents an article, we must first acquire sentence level data structures. The sentences need to be further aggregated together to create an article. Let's consider the sentences in Figure 3.8 as illustrative of an article.

> The negotiations on weapon management are failing. This is concerning, as failure to keep the parties at the negotiating table will result in illicit weapons on the black market.

Figure 3.8: Original article to be analyzed

As evident, the general theme being discussed centers around a set of negotiations regarding weapons and the potential consequences of a failure in the negotiations. It is imperative that our data structure represent this. Examining each sentence individually, we get the sets of tuples per sentence shown in Figure 3.9.

The negotiations on weapon management are failing:

```
[negotiations:negotiations.n, weapon:weapon.n,
management:management.n-u, failing:failing.g]
```

This is concerning, as failure to keep the parties at the negotiating table will result in illicit weapons on the black market:

```
[concerning:concerning.g, failure:failure.n,
parties:parties.n, negotiating:negotiating.g,
table:table.n, illicit:illicit.a, weapons:weapons.n,
black:black.a]
```

Figure 3.9: Tuples per sentence

We've picked out the words "negotiations" and "negotiating". Similarly, we've also extracted "weapon" and "weapons". While we know that these *mean* the same thing, we have to normalize these words to some common form so that we can create a data structure that conveys that negotiations and weapons are thematic in the two sentences. To do this, we turn to a concept called Stemming.

3.2.3 Stemming

Aggregating frequencies of words isn't equivalent to aggregating frequencies of numbers. Words may inherently mean the same thing but be spelled differently as a result of plural/singular or other differences. Consider the following words that effectively convey the same theme but are spelled differently according to their tense, or other context:

dismiss, dismissed, dismissal, dismisses, dismissing

From a string comparison standpoint, none of the above words are identical. Yet, their base, or root word all trace back to "dismiss". The context, whether used as an active verb, a verb in the past or future tense, or others have no bearing on what the core of each word conveys. To reduce all these variants down to their base form, we can employ a stemming algorithm.

Stemming is a mechanism that allows us to analyze a word and either strip or re-suffix its latter part such that it is "stemmed" down to a base form. In some cases, the base form may not be an actual word, but this is acceptable as long as the word reduction is deterministic and accurate. A common, opensource stemming algorithm is the Porter Stemming[10] technique. This is widely documented, with implementations in numerous languages. Running the algorithm over the variations of "dismiss", we get the following output:

Original Word	Stemmed Word
dismiss	dismiss
dismissed	dismiss
dismissal	dismiss
dismisses	dismiss
dismissing	dismiss

Figure 3.10: Variations of the word "dismis" with their stemmed equivalents

As evident, the stemming algorithm allows us to reduce all the variations of a word down to a common base form. Typically, a stemmer will look for and remove sequences of alphabets that suffix a word. Sometimes a

stemmer will replace a sequence with a shorter one (eg: economies may become economi). Sufficeth to say that the use of a stemmer is invaluable as it allows us to combine words used within and across articles that are merely variations of one another to one common base form. In turn, this allows us to analyze words across sentences and articles that may not have been written in the same tense, or grammatical context, but still convey the same meaning.

3.2.4 Article Histogram

The most rudimentary data structure for an article is one that identifies frequencies of words aggregated up from a sentence. Effectively, a histogram of words within the article. The value in this data structure is its ability to quickly identify the set of core concepts that the article is relevant to. It also allows us to create a fingerprint of what the article contains. We can normalize a histogram and compare one to another, allowing us to group various articles together that may be dealing with the same or similar concepts.

The mechanism to do this is fairly straightforward. Each article histogram is merely a map whose keys are stemmed words, and whose corresponding values are the frequencies of the stemmed words within the article, across sentences. Let us revisit the example article whose sentence level tuples are illustrated in Figure 3.9. Stemming each word in the set of tuples, we get the data in Figure 3.11:

Iterating through all of the pairs (from Figure 3.11), we can use the

Word as seen in sentence	Stemmed word
negotiations	negoti
weapon	weapon
management	manag
failing	fail
concerning	concern
failure	failur
parties	parti
negotiating	negoti
table	tabl
illicit	illicit
weapons	weapon
black	black

Figure 3.11: Stemmed words from a link-parsed article

stemmed word as a key to two values. First, we are interested in frequency of occurrence within the article. Second, we are interested in the forms that a word or term occurred within the sentence, which forms a list where we have frequencies greater than one of a stemmed word. The end result for our example article is show in tabular form in Figure 3.12, and rendered as a pie chart in Figure 3.13. Figure 3.14 provides a view of an actual article as redered in a pie-chart. This article histogram data structure forms the core construct of how we decompose an article. It also makes for easy durability, which is what we broach next.

Stemmed Word	Frequency	Occurred in Article as
negoti	2	negotiating negotiations
weapon	2	weapon weapons
black	1	black
concern	1	concerning
fail	1	failing
failur	1	failure
illicit	1	illicit
manag	1	management
parti	1	parties
tabl	1	table

Figure 3.12: An article histogram

Figure 3.13: Depicting Figure 3.12 via a Pie chart

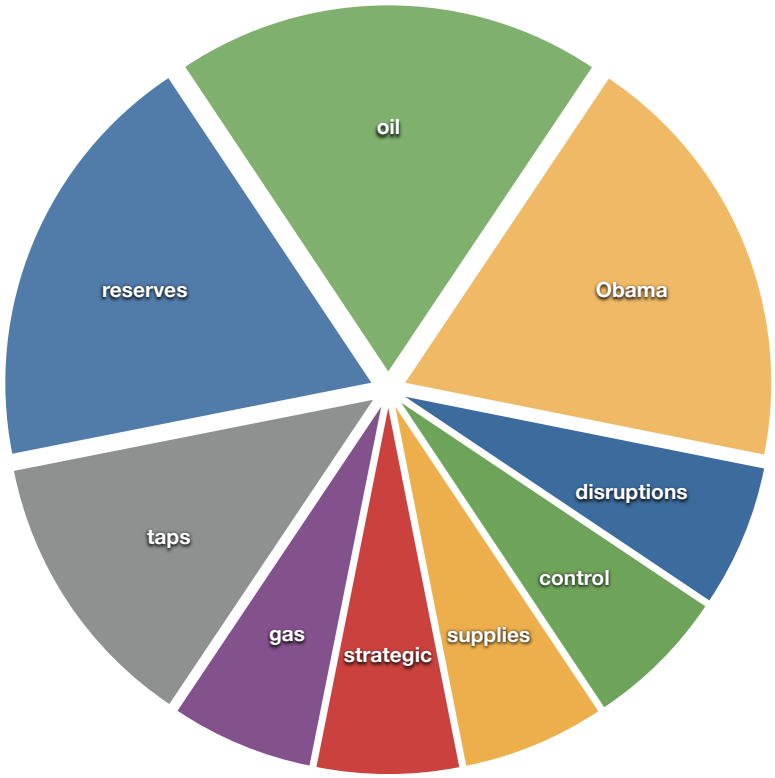

Figure 3.14: Pie representation of article histogram from analysis of "Obama prepared to tap oil reserves" on CNN Money, March 11, 2011

Chapter 4

A Database to Support Analysis

An article histogram provides us with article level insight for a specific article, from a specific source, at a specific point in time. To be able to look across various vectors like time, news providers, other articles, and by terms, we need to be able to look at many article histograms at once. Additionally, we need to be able to provide durability to the data structures. To support this, we opted to create a relational database to provide to persist the article histograms to.

In addition to solving durability and providing a queryable store for the article histograms, a database also allowed for managing configuration of news feeds and a place to identify and flag terms that weren't meaningful due to their prevalence.

The rest of this chapter covers the structure of the database and its interactions with the news acquisition and analysis portions, as well as a review on the sorts of queries it supports.

4.1 Schema

The implemented schema incorporated concepts that we hoped to be able to use to add more functionality in the future. Principally, we assume that news is delivered from a source. As an example, Reuters and the Associated Press are both sources. We get articles via a source, which in turns provides an article histogram. Orthogonal to the above linear flow are words or terms. These exist across all articles, histograms and time. The described schema's tables mirror this thought process.

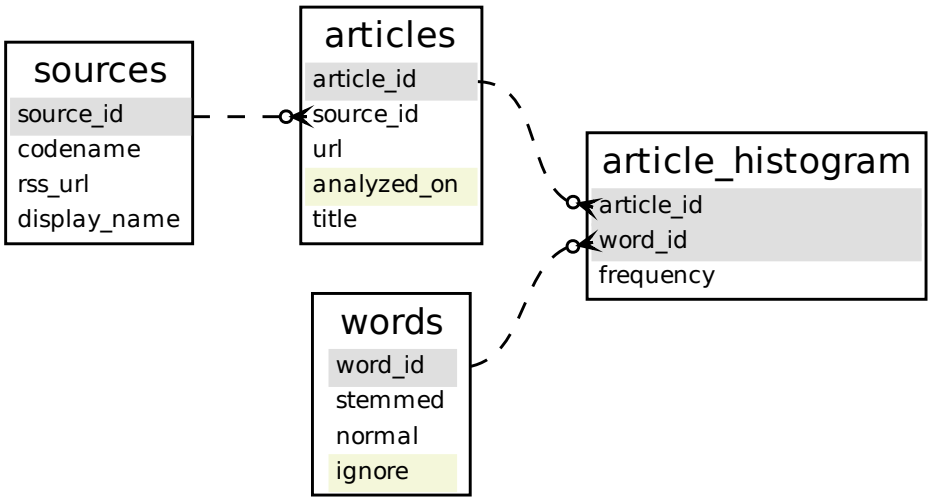

Figure 4.1: Schema for a database to store article histograms

4.1.1 Sources

Being able to categorize data by a source has value when you want to compare and contrast the news delivered by varying sources. As an example, having a source level categorization allows us to find all articles about a specific term and then partition the results or calculations thereof by corresponding source. An additional benefit of describing a source is that we can associate information at this level about how to acquire news for the specific source (the RSS feed in our current framework). The established table has the following columns:

- source id, the primary key for the table,
- codename, a short descriptive name that is used in debugging or to provide as a hook to a specific source in lieu of a source id,
- rss url, the RSS feed URL that the new acquisition components require to fetch content,
- display name, a display friendly name used in reporting.

4.1.2 Articles

Articles are what a source provides. We don't store the content of an article, but we do store a pointer to the original article via a URL. Additionally, we store a timestamp that is part of the RSS published article that positions the article in time. The table breaks down into the following columns:

- article id, the primary key for the table,
- source id, the source from which this article was sourced,
- url, to the article,
- analyzed on, that represents the date and time that the article was published,
- title, that the article was published with.

4.1.3 Words

The words table stores terms, which are frequently words. Since we want to have the ability to scan for words across articles and time, this table serves as a reference for all words/terms that we have encountered as a canonical reference. Some words tend to pollute our reporting abilities because they are prevalent across all articles and don't add any inherent meaning or value that distinguishes their use in an article. As such, we qualify these as stop-words, or noise. Examples of such words/phrases include "Read more", or the name of a news source that frequently occurs at the beginning of an article. Our techniques to identify words to ignore are discussed below.

- word id, the primary key for the table,
- stemmed, a stemmed version of the word,
- normal, a presentable version of the word,

- ignore, whether the word should be ignored from normal reporting.

Stop Words and Terms: Ignorable

We needed an automated way to identify stop words, or words to be ignored within our body of words. There are some existing techniques that are quite effective at this. The concepts with Term Frequency/Inverse Document Frequency (TF/IDF) are applicable to this scenario. Typically TF/IDF associates uniqueness and relevance of terms within a document across an entire corpus. Within the scope of the current endeavor, running a TF/IDF strategy across articles, within the scope of a given Source will neutralize terms used heavily within a source, but allow for them to show up as relevant across sources. [1]

4.1.4 Article Histogram

The Article Histogram is the core table within the schema. It contains the article histogram data structure, providing per article term frequencies. The composition of the table is quite simple:

- article id, that this word or term is associated with (one half of the primary key),

[1] While TF/IDF is an appealing strategy, it wasn't implemented at the time this document was authored. Our current strategy is a simple standard deviation based model that is interactive, and run on an ad-hoc basis.

- word id, the specific word/term being measured (the other half of the primary key),
- frequency, and its corresponding frequency within the article.

4.2 Capabilities

4.2.1 Prominent Words in a Timeframe

To identify the set of words/terms that are notable within a span of time, a query can limit the set of articles analyzed to a time boundary. We further exclude words that have been flagged as ignorable, and finally join the articles and words to article histograms. Figure 4.2 illustrates a query and resulting data scoped for March 2011. As evident, during the month of March, a significant amount of news centered around the Japanese earthquake and subsequent nuclear plant problems, as well as the protests in Libya.

4.2.2 Related Terms

Given a specific word or term, and a time boundary, the schema allows us to inquire about the set of related words and terms. As an example that builds from Figure 4.2, the data in Figure 4.3 shows the set of terms that are related to the word "Japan" in March. As evident, the related words are reflective of the earthquake and following nuclear plant crisis.

```
select w.word_id, w.normal, sum(wh.frequency) as count
  from words w, articles a, word_histogram wh
  where w.ignore = 0 and
  a.analyzed_on > '2011-03-01' and
  a.analyzed_on < '2011-03-31' and
  wh.article_id = a.article_id and
  wh.word_id = w.word_id
  group by wh.word_id order by count desc limit 10;
```

word id	normal	count
1156	Japan	1955
4625	Libya	1179
2117	nuclear	980
267	power	620
688	plant	602
4426	earthquake	509
103	forces	456
1204	Japanese	419
3883	reactor	412
686	protest	360

Figure 4.2: Query and resulting data for the top ten words/terms of March

4.2.3 Analysis by Outlet

A slight variation on the related term search allows us to partition the data by articles from a specific source. Continuing from the analysis of "Japan" in March, Figure 4.4 shows two output sets based on two different news outlets. The words from The Wall Street Journal reflect a focus on words like "relief", "damaging", and "humanitarian", while the some of the differing words from the Associated Press are "threat", "system", "agency".

```
select w.word_id, w.normal, sum(wh2.frequency) wcount
  from words w, word_histogram wh, word_histogram wh2, articles a
  where wh.word_id = 1156 and
  wh.article_id = a.article_id
  and wh2.article_id = wh.article_id
  and w.word_id = wh2.word_id
  and w.ignore = 0 and wh2.word_id != wh.word_id
  and a.analyzed_on > '2011-03-01'
  and a.analyzed_on < '2011-03-31'
  group by w.word_id order by 3 desc limit 10;
```

word id	normal	count
2117	nuclear	753
688	plant	426
4426	earthquake	415
3883	reactor	335
267	power	266
1204	Japanese	257
905	radiation	232
455	crisis	181
463	disaster	181
70	safety	113

Figure 4.3: Related words for "Japan" in March 2011

```
select w.word_id, w.normal, sum(wh2.frequency) wcount
  from words w, word_histogram wh, word_histogram wh2, articles a, sources s
  where wh.word_id = 1156 and
  wh.article_id = a.article_id
  and wh2.article_id = wh.article_id
  and w.word_id = wh2.word_id
  and w.ignore = 0 and wh2.word_id != wh.word_id
  and a.analyzed_on > '2011-03-01'
  and a.analyzed_on < '2011-03-31'
  and a.source_id = s.source_id
  and s.source_id = ?
  group by w.word_id order by 3 desc limit 10;
```

From "The Wall Street Journal"		
word id	normal	count
2117	nuclear	110
688	plant	65
4426	earthquake	63
455	crisis	31
649	relief	28
3883	reactor	26
3328	damaging	25
6426	humanitarian	25
1204	Japanese	25
905	radiation	23
From "Associated Press"		
word id	normal	count
2117	nuclear	20
455	crisis	12
4426	earthquake	12
905	radiation	11
688	plant	6
463	disaster	5
1320	amount	5
1675	agency	4
275	system	4
551	threat	4

Figure 4.4: Related words for "Japan" in March 2011, split by two different sources.

Chapter 5

Key Takeaways

We created a wordsinmedia.com as a concrete implementation for the feed acquisition and article processing system. Additionally, an HTML5 front-end demonstrates various reporting possibilities that are capable over the described schema. We encourage you to examine the website and interact with it. One of its goals was to present news in an unbiased, non-editorialized context and we think we have made positive gains in that direction. There were some significant learnings as we created the site and we felt them significant to capture. Additionally, there are many avenues that we feel are viable future areas for additional research that are also important to identify.

5.1 Lessons Learned

5.1.1 RSS Limitations

As documented in Chapter 2, RSS is encumbered by issues that complicate its processing. Advertisements, sponsored links, in-article references to other articles, menus and various other presentation artifacts make it non trivial to extract the content of the article. However, a much more severe problem is steadily growing: most news outlets have started to truncate article content provided via RSS feeds, resulting in a feed only providing the first few paragraphs of content.

One mechanism (which was not implemented) would be to follow the RSS item's link to the full content of the article and recover it via an in-memory browser like HTMLUnit that renders content to a string. This is unfortunately not viable in the long term either as news outlets place paywalls that limit the amount of content available. While there are ways to circumvent this too, the ethicality of such actions is questionable.

5.1.2 Sentence Fracturing

Extracting sentences from an article is another problem fraught with numerous exceptions and failure conditions. A few common examples include situations where an article contains quoted dialog like so:

> After pausing briefly, he said "it would be hard to consider such

> an action. But there may be scope in the future," and walked out.

Additional complexity includes abbreviations on words that aren't frequently abbreviated, website addresses and such. Rendering issues where line terminations are added increase complexity too. Sometimes it makes sense to strip the newlines and replace them with periods, while in other cases, spaces may be more appropriate.

5.1.3 Performance

Examining a sentence with Link Grammar is a performance intensive operation. Yet, long sentences are quite common in the news and with a link parser configured to aggressively scan a sentence for all possible linkages, each parse can consume a single CPU for many seconds. If an article contains a mere ten sentences, you could spend about a minute processing the article. Using a single CPU, you are likely to fall behind over the course of an hour if more than 60 articles are to be scanned.

A simple model to create a much more scalable analysis layer would be to enqueue articles to be scanned into a distributed queue, and to have numerous analysis engines on different CPUs consume transactionally from the queue. This would allow for elasticity at the analysis layer which could be automated based on the queue size. As such, under load, additional analysis nodes could be dynamically provisioned and released when the queue size reduces.

5.1.4 Schema

The schema, while efficient should have had a few tweaks to make it a lot more efficient and more powerful. Foremost, the stop word flag should have been at the word histogram level and not globally scoped at the word level. The global scope prevents us from allowing in common words like 'Fox News', in the event that the news outlet itself (which is globally marked as a stop word) has made it into a story. Having the stop word identification at the article histogram allows for this to be managed per article and/or source.

Additionally, the source id per histogram could be stored within the article histogram table too, changing the schema to be more fact/dimensional in nature than currently so. This would allow per source scoping very easily, as opposed to the current join required on sources, articles, and then to article histogram. In similar vein, the schema should have provisioned for storing or representing nouns and descriptive words. This would have allowed for extraction of thematic versus factual references without having to post-process words to see which bucket they fall in.

5.2 Future Research Areas

This thesis puts forth some basic concepts that can be leveraged to allow for an aggregated, factual and real time view of the news. However, this can serve as a foundation for a much deeper analysis into the polarity of various outlets per subject or theme. A lot more data would have to be

stored and made available for analysis to execute this, though storage and compute power are readily available.

5.2.1 Sentence Fragmentation

Some fundamental problems that need to be examined are sentence fragmentation issues. One strategy may be to provide different articles to different sentence fracturing treatment strategies and to see which ones tend to perform the best. Performance can be measured in conjunction with how effectively the link parser is able to execute successfully.

5.2.2 Sentence Parsing

The link parser too should be split into multiple treatments with varying calibrations. Additionally, there are other parsers that execute similar sentence decomposition strategies that can provide the same information as the link parser. Various treatments could adopt various analysis strategies and we could calibrate on highly successful ones.

5.2.3 Synonyms

Finally, the corpus of stored data could be compressed and made much more potent if synonyms were clubbed together in a lossless fashion. Two articles describing the “arrest” or “apprehension” of a subject are effectively referring to the same theme. WordNet and other mechanisms provide ways to make

such associations that can be used to aggregate a lot more article data in a thematic fashion.

Chapter 6

Conclusion

While aggregating unstructured data is inherently complex, it is possible. Extracting specific factual data from the news is well at hand. Using an amalgamation of simple techniques, it is possible to create representative factual term graphs that accurately convey events and their stories without editorial bias.

The potential applications for such a concept are wide. Beyond the news, this can be used to aggregate and accurately portray collected verbatim data for market research, clinical or other studies. Further, this technique can be applied to other areas where textual data is provided in real time, especially amid social media landscapes.

Appendix A

Existing News Aggregators

There are numerous solutions that aggregate news in specific ways today. Google News, bing News, Yahoo! News and others provide mechanisms to scan and drill in to large volumes of articles. All of these mechanisms retain the exact wording and content of the original news article, and provide value by creating groupings of thematically homogenous content within a temporal context. In addition, drill-down operations typically allow the user to see specific articles. A variation on this pattern is offered by newsmap, which leverages Google News to provide key headlines arranged in a Treemap based on the relevant article counts for a story. These solutions tend to provide quick ways to get to articles, but don't offer the user an ability to analyze the news, as we'd expect a user to analyze a spreadsheet of financial data, or traffic to a web server.

Appendix B

Random Statistics

Based on data gathered from analysis between August 2010 and April 2011.

- 76,946 articles scanned
- 35,606 words and terms identified
- 758,348 data points in resulting histogram
- News outlets scanned:
    1. Time
    2. Fox News
    3. ABC
    4. CBS
    5. National Public Radio
    6. New York Times
    7. CNN
    8. Reuters
    9. Associated Press
    10. Huffington Post

11. Washington Post
12. Wall Street Journal

- Top ten words across all time and sources:

1. Japan
2. Libya
3. protest
4. nuclear
5. power
6. forces
7. cut
8. Egypt
9. oil
10. planning

Appendix C

Journalism

C.1 Comprehensive Coverage vs. Publisher Interests

News coverage runs very wide covering topics like science, history, finance, automobiles, entertainment, politics, and a lot more. Most of our news is delivered by specific organizations (eg: CNN, Fox News, New York Times, Washington Post) that all have to generate revenue. The lifeblood for these organizations is via advertising [13]. Advertisers look for specific demographics to advertise to. It is therefore in the best interests of a news outlet to identify a section of population to engage with their journalism. This segment may be geographic, or politically oriented, or may have specific interests (eg: financial news), or any other attribute. The net result is that a news outlet is most likely to provide the news that their current reader base will be interested in so that it can maintain its loyalty base. This runs at odds with

comprehensiveness since doing so is likely to jeopardize readership. How can someone interested in the news be sure that they are getting the coverage that they desire?

C.2 Accuracy and Integrity Issues in Journalism

Journalists are free to leverage specific quotes to shape a news article in a specific way. They may choose to empathize via specific adjective placements. Specific topics that may be related might be highlighted, or ignored in the course of an article. Given a wide array of articles about the same subject, a reader is likely to overcome this issue, but this would be a very time consuming proposition, and in some cases prohibitively so. More gratuitous journalistic errors are present in recent history [14]. Given the volumes in question, and known journalism controversies, how can a reader be assured that the select few sources they trust are accurate?

Appendix D

Source Code

D.1 Database Schema

Listing D.1: "Schema generation SQL"

```
--
-- Table structure for table `sources`
--
DROP TABLE IF EXISTS `sources`;
CREATE TABLE `sources` (
 `source_id` int(11) NOT NULL AUTO_INCREMENT,
 `codename` varchar(64) NOT NULL,
 `rss_url` varchar(1024) NOT NULL,
 `display_name` varchar(512) NOT NULL,
 PRIMARY KEY (`source_id`)
) ENGINE=MyISAM AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;

--
-- Table structure for table `articles`
--
DROP TABLE IF EXISTS `articles`;
CREATE TABLE `articles` (
`article_id` int(11) NOT NULL AUTO_INCREMENT,
`source_id` int(11) NOT NULL,
`url` varchar(2048) NOT NULL,
`analyzed_on` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
    CURRENT_TIMESTAMP,
`title` varchar(512) DEFAULT NULL,
PRIMARY KEY (`article_id`),
```

```
KEY `i_analyzed_on` (`analyzed_on`)
) ENGINE=MyISAM AUTO_INCREMENT=79743 DEFAULT CHARSET=utf8;

--
-- Table structure for table `words`
--
DROP TABLE IF EXISTS `words`;
CREATE TABLE `words` (
 `word_id` int(11) NOT NULL AUTO_INCREMENT,
 `stemmed` varchar(1024) NOT NULL,
 `normal` varchar(1024) NOT NULL,
 `ignore` bit(1) NOT NULL DEFAULT b'0',
 PRIMARY KEY (`word_id`),
 KEY `i_ignore` (`ignore`)
) ENGINE=MyISAM AUTO_INCREMENT=36811 DEFAULT CHARSET=utf8;

--
-- Table structure for table `word_histogram`
--
DROP TABLE IF EXISTS `word_histogram`;
CREATE TABLE `word_histogram` (
  `article_id` int(11) NOT NULL,
  `word_id` int(11) NOT NULL,
  `frequency` int(11) NOT NULL,
  PRIMARY KEY (`article_id`,`word_id`),
  KEY `i_wh_word_id` (`word_id`),
  KEY `i_wh_article_id` (`article_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

D.2 RSS Acquisition and Analysis

Listing D.2: src/org/recursed/w/server/rss/AnalysisSupport.pm

```
use XML::FeedPP;
use HTML::TreeBuilder;
use HTML::FormatText;
use Lingua::Sentence;

use PorterStemmer;
use WordNetSupport;
use LinkParserSupport;

use strict;
use warnings;

sub extractText {
 my $chunk = shift;
 my $tree = HTML::TreeBuilder->new_from_content($chunk);
 #$tree->parse($chunk);
 my $formatter =
   HTML::FormatText->new( leftmargin => 0, rightmargin => 50000000 );
 my $result = $formatter->format($tree);
 $tree->delete();
```

```
 # remove extranneous [IMAGE] and such tags
 #print "Formatted: \n$result\n";
 $result =~ s/\[.*\]//g; # this removes the [IMAGE] and other tags
 $result =~ s/[ \t]+\*.+\n/ /g; # eliminate the trailing bullets in foxnews
 $result =~ s/Email this Article.+//g;
 $result =~ s/\n/ /g; # eliminate newlines and make them delimited sentences.
  return $result;
}

sub getItems {
 my $source = shift;
 my $feed = XML::FeedPP->new($source);
 return $feed->get_item();
}

my $splitter = Lingua::Sentence->new("en");

sub analyze {
 my $content = shift;
 my %articleHash = ();
 my %articleWords = ();
 #print "\n Going to try and parse: \n $content\n";
 my @sentences = split( /\n/, $splitter->split($content) );
 my $hadErrs = 0;
 foreach my $sentence (@sentences) {
  my %wordsOfInterest = get_words_of_interest($sentence);
  if (scalar(%wordsOfInterest) eq "0") {
   print "!";
   $hadErrs = 1;
   # we failed to run the linker..
   # manually add words via WordNet..
   my @words = split(/\s/, $sentence);
   my $numWords = @words;
   for my $examine (@words) {
    if (lookupWord($examine) == 1) {
     $wordsOfInterest{$examine} = $examine;
    }
   }
  } else {
   print ".";
  }
  # At this point, we should have the sentence hash,
  # from the Linker, or from WordNet.
  # We now iterate through each word returned,
  # and add it to the a hash articleWords that maintains
  # a histogram of word frequencies.
  # Additionally, we hold store references of the word in
  # articleWords.
  for my $key (keys(%wordsOfInterest)) {
   my $word = stem(lc($key));
   $articleWords{$word}{$key}="a";
   my $currentValue = $articleHash{$word};
   if (defined($currentValue)) {
    $currentValue++;
   } else {
    $currentValue = 1;
   }
```

```
   $articleHash{$word} = $currentValue;
  }
 }
 print "\n";
 my %analysis = ();
 $analysis{histogram}=\%articleHash;
 $analysis{words}=\%articleWords;

 if ($hadErrs == 1) {
  print "PARSE_FAIL_IN: \n$content\n";
 }
 return \%analysis;
}

#my $content = "A sheriff's deputy investigating a report of gunfire at a trailer park was shot dead
      Saturday, and the shooting suspect was killed after a furious standoff with police, authorities said.
      A police officer was wounded.";
#my $analysis_ref = analyze($content);
#my %analysis = %$analysis_ref;
#my %articleHash = %{$analysis{histogram}};
#my %articleWords = %{$analysis{words}};
#print "\nArticle Histogram:\n";
#for my $key (keys %articleHash) {
# print $key, "(", $articleHash{$key}, ") for: ";
# for my $word (keys %{$articleWords{$key}}) {
# print $word, " ";
# }
# print "\n";
#}

1;
```

Listing D.3: src/org/recursed/w/server/rss/DatabaseSupport.pm

```
use DBI;
use Source;
use Word;

use strict;
use warnings;

my $connection = "DBI:mysql:database=w;host=localhost";
my $username = "";
my $password = "";
my %properties = ('RaiseError' => 1);

# Connect to the database.

sub get_sources {
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("SELECT source_id, codename, rss_url FROM sources");
 my @sources = ();
 $sth->execute();
 while (my @array = $sth->fetchrow_array()) {
  my $source = new Source(@array);
```

```
  push (@sources, $source);
 }
 $sth->finish();
 $dbh->disconnect();
 return @sources;
}

sub is_article_analyzed {
 my $articleURL = shift;
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("SELECT count(*) FROM articles where url = '$articleURL'");
 my $count;
 $sth->execute();
 while (my @array = $sth->fetchrow_array()) {
  $count = $array[0];
 }
 $sth->finish();
 $dbh->disconnect();
 return $count;
}

sub insert_article_analysis {
 my $source = shift;
 my $title = shift;
 my $article_url = shift;
 # insert record in articles
 my $article_id = insert_article_record($source->{id}, $article_url, $title);
 my $analysis_ref = shift;
 my %analysis = %$analysis_ref;
 my %articleHash = %{$analysis{histogram}};
 my %articleWords = %{$analysis{words}};
 # print "Got an article for ", $source->{codename}, "\n Article URL: ", $article_url, "\n";
 # print " Article Histogram:\n";
 for my $key (keys %articleHash) {
# print " ",$key, "(", $articleHash{$key}, ") for: ";
  my @normal_representations = keys %{$articleWords{$key}};
  my $first_normal = $normal_representations[0];
  my $word_id = reconcile_word($key, $first_normal);
  append_histogram($article_id, $word_id, $articleHash{$key});
# for my $word (keys %{$articleWords{$key}}) {
# print $word, " ";
# }
# print "\n";
 }
}

sub insert_article_record {
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("insert into articles (source_id, url, title) values (?,?,?)");
 $sth->execute(shift, shift, shift);
 $sth->finish();
 my $article_id = $dbh->last_insert_id(undef, undef, "articles", undef);
 $dbh->disconnect();
 return $article_id;
}

sub reconcile_word {
```

```
 my $stemmed = shift;
 my $normal = shift;
 my $word_id = undef;
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $lookup = $dbh->prepare("select word_id from words where stemmed = ?");
 $lookup->execute($stemmed);
 while (my @array = $lookup->fetchrow_array()) {
  $word_id = $array[0];
 }
 $lookup->finish();
 if (!defined $word_id) {
  my $insert = $dbh->prepare("insert into words (stemmed, normal) values (?,?)");
  $insert->execute($stemmed, $normal);
  $insert->finish();
  $word_id = $dbh->last_insert_id(undef, undef, "words", undef);
 }
 $dbh->disconnect();
 return $word_id;
}

sub append_histogram {
 my $article_id = shift;
 my $word_id = shift;
 my $frequency = shift;
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $insert = $dbh->prepare("insert into word_histogram (article_id, word_id, frequency) values
      (?,?,?)");
 $insert->execute($article_id, $word_id, $frequency);
 $insert->finish();
 $dbh->disconnect();
}

sub get_all_words {
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("select w.word_id, w.normal, sum(wh.frequency) as sf from words w,
      word_histogram wh where wh.word_id = w.word_id group by wh.word_id order by sf asc");
 my @words = ();
 $sth->execute();
 while (my @array = $sth->fetchrow_array()) {
  my $w = new Word(@array);
  push (@words, $w);
 }
 $sth->finish();
 $dbh->disconnect();
 return @words;
}

sub mark_as_ignorable {
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("update words w set w.ignore = 1 where w.word_id = ?");
 $sth->execute(shift);
 $sth->finish();
 $dbh->disconnect();
}

sub get_non_summarized_dates {
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
```

```
 my $sth = $dbh->prepare("select
       date(a.analyzed_on) an_dates
       from
       articles a
       where
       date(a.analyzed_on) not in
       (
       select date(sd.summarized_on) sd_found from summary_days sd
       group by sd_found
       order by sd_found asc
       )
       and date(a.analyzed_on) != date(now())
       group by an_dates
       order by an_dates asc
       ");
 my @result;
 $sth->execute();
 while (my @array = $sth->fetchrow_array()) {
  push(@result, $array[0]);
 }
 $sth->finish();
 $dbh->disconnect();
 return @result;
}

sub etl_summary_day {
 my $date_to_summarize = shift or die "no date supplied";
 my $dbh = DBI->connect($connection, $username, $password, \%properties);
 my $sth = $dbh->prepare("insert into summary_days (summarized_on, word_id, frequency)
          select
          date(a.analyzed_on), wh.word_id, sum(wh.frequency)
          from
          word_histogram wh,
          articles a,
          words w
          where
          wh.article_id = a.article_id
          and date(a.analyzed_on) = ?
          and w.word_id = wh.word_id
          and w.ignore = 0
          group by wh.word_id
          order by 2, wh.word_id");
 $sth->execute($date_to_summarize);
 $sth->finish();
 $dbh->disconnect();
}

1;
```

Listing D.4: src/org/recursed/w/server/rss/LinkParserSupport.pm

```
use warnings;
use strict;

use Lingua::LinkParser;
```

```
sub get_words_of_interest {
 my $t1 = time;
 my %wordsOfInterest = ();
 my @nonParseableSentences = ();
 my $sent = shift;

 # this is a very simple definition that I see in use in the examples
 # I would like to move to just this one definition instead of the numerous
 # types with various panic modes and such..
 my $parser = new Lingua::LinkParser(verbosity => 0, display_on => 0);
 $parser->opts(
   'max_null_count' => 3,
   'min_null_count' => 1
 );
 my $sentence = $parser->create_sentence($sent);
 if ( $sentence->num_linkages == 0 ) {
  $parser->opts(
   'min_null_count' => 1,
   'max_null_count' => $sentence->length
  );
  $sentence = $parser->create_sentence($sent);
  if ( $sentence->num_linkages == 0 ) {
   $parser->opts(
    'disjunct_cost' => 3,
    'min_null_count' => 1,
    'max_null_count' => 30,
    'max_parse_time' => 60,
    'islands_ok' => 1,
    'short_length' => 6,
    'all_short_connectors' => 1,
    'linkage_limit' => 100
   );
   $sentence = $parser->create_sentence($sent);
  }
 }


 my @bigStruct = $sentence->get_bigstruct;


 my $wordCount = @bigStruct;
 foreach my $i ( 1 .. $wordCount ) {
  my $word = $bigStruct[$i]->{word};

  # April 10th: This is a much cleaner looking way to pull words.
  # I think the code after this while loop needs to be refactored to be placed
  # in this loop, and use the links a lot more. Also, note that the first pass
  # parser is a lot simpler than prior. This may have significant speedup over
  # the prior implementation with all sorts of switches to the config.
  while (my ($k,$v) = each %{$bigStruct[$i]->{links}} ) {
    # Linkages of type G are typically nouns like names
    # We want to pull them in only one directional sequence, hence the extra check
    if ($k =~ m/G.*/ && $i<$v) {
      my $concatWord = $sentence->get_word($i)." ".$sentence->get_word($v);
      my $rawWord = $word." ".$bigStruct[$v]->{word};
      $wordsOfInterest{$concatWord}=$rawWord;
```

```
      print "Found a concat: ".$concatWord, "\n";
    }
  }

 #while ((my $linkageType, my $wordID) = each %{$bigStruct[$i]->{links}}){
 # print $word, " is linked by ", $linkageType, " to ", $bigStruct[$wordID]->{word}, "\n";
 #}


  if ( defined($word) && $word =~ m/.*\.[a,b,g,m,n,l,f,g].*/ ) {
   #print $word, "\n";

   if ($word =~ m/.*\.[f].*/) {
    #TODO: We want to expand this usage to test for proper nouns using the G connector recursively.
    # Eg:
    # +----G---+----G----+---AN---+---Ss--+-Pp-+
    # | | | | | |
    # George.b Herbert.m Walker.m Bush[?].n is.v here
    #
    # this is a noun.. testing for AN linkages
    while ((my $linkageType, my $wordID) = each %{$bigStruct[$i]->{links}}){
     if ($i == $wordID - 1) {
      my $compositeWord = $sentence->get_word($i). " ". $sentence->get_word($wordID);
      #print $sentence->get_word($i), " ", $sentence->get_word($wordID), "\n";
      $wordsOfInterest{$compositeWord} = $word;
     }
     #print $word, " is linked by ", $linkageType, " to ", $bigStruct[$wordID]->{word}, "\n";
    }
   }
   my $wordOfInterest = $sentence->get_word($i);
   $wordsOfInterest{$wordOfInterest} = $word;
  }
 }

 my $t4 = time;

 #print "Big struct: ", ($t4 - $t3), "s\n";

 return %wordsOfInterest;
}

#my $content = "The Obama administration is monitoring and planning its next move, with embassy
      personnel still in harm's way and attempts to reach Gbagbo unsuccessful.";
#my $content = "That morning newspapers carried the story of John Gallianos dismissal from Dior.";
#my %map = get_words_of_interest($content);
#foreach my $key (keys %map) {
# print $key, ", ", $map{$key}, "\n";
#}

1;
```

Listing D.5: src/org/recursed/w/server/rss/PorterStemmer.pm

```
# Adapted from the PorterStemmer perl example:
#
```

```
# Porter stemmer in Perl. Few comments, but it's easy to follow against the rules in the original
# paper, in
#
# Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
# no. 3, pp 130–137,
#
# see also http://www.tartarus.org/~martin/PorterStemmer

# Release 1

use warnings;
use strict;

my %step2list;
my %step3list;
my ($c, $v, $C, $V, $mgr0, $meq1, $mgr1, $_v);


sub stem
{ my ($stem, $suffix, $firstch);
   my $w = shift;
   if (length($w) < 3) { return $w; } # length at least 3
   # now map initial y to Y so that the patterns never treat it as vowel:
   $w =~ /^./; $firstch = $&;
   if ($firstch =~ /^y/) { $w = ucfirst $w; }

   # Step 1a
   if ($w =~ /(ss|i)es$/) { $w=$`.$1; }
   elsif ($w =~ /([^s])s$/) { $w=$`.$1; }
   # Step 1b
   if ($w =~ /eed$/) { if ($` =~ /$mgr0/o) { chop($w); } }
   elsif ($w =~ /(ed|ing)$/)
   { $stem = $`;
     if ($stem =~ /$_v/o)
     { $w = $stem;
       if ($w =~ /(at|bl|iz)$/) { $w .= "e"; }
       elsif ($w =~ /([^aeiouylsz])\1$/) { chop($w); }
       elsif ($w =~ /^${C}${v}[^aeiouwxy]$/o) { $w .= "e"; }
     }
   }
   # Step 1c
   if ($w =~ /y$/) { $stem = $`; if ($stem =~ /$_v/o) { $w = $stem."i"; } }

   # Step 2
   if ($w =~ /(ational|tional|enci|anci|izer|bli|alli|entli|eli|ousli|ization|ation|ator|alism|
       iveness|fulness|ousness|aliti|iviti|biliti|logi)$/)
   { $stem = $`; $suffix = $1;
     if ($stem =~ /$mgr0/o) { $w = $stem . $step2list{$suffix}; }
   }

   # Step 3

   if ($w =~ /(icate|ative|alize|iciti|ical|ful|ness)$/)
   { $stem = $`; $suffix = $1;
     if ($stem =~ /$mgr0/o) { $w = $stem . $step3list{$suffix}; }
   }
```

```
    # Step 4

    if ($w =~ /(al|ance|ence|er|ic|able|ible|ant|ement|ment|ent|ou|ism|ate|iti|ous|ive|ize)$/)
    { $stem = $`; if ($stem =~ /$mgr1/o) { $w = $stem; } }
    elsif ($w =~ /(s|t)(ion)$/)
    { $stem = $` . $1; if ($stem =~ /$mgr1/o) { $w = $stem; } }


    # Step 5

    if ($w =~ /e$/)
    { $stem = $`;
      if ($stem =~ /$mgr1/o or
          ($stem =~ /$meq1/o and not $stem =~ /^${C}${v}[^aeiouwxy]$/o))
         { $w = $stem; }
    }
    if ($w =~ /ll$/ and $w =~ /$mgr1/o) { chop($w); }

    # and turn initial Y back to y
    if ($firstch =~ /^y/) { $w = lcfirst $w; }
    return $w;
}

sub initialise {

    %step2list =
    ( 'ational'=>'ate', 'tional'=>'tion', 'enci'=>'ence', 'anci'=>'ance', 'izer'=>'ize', 'bli'=>'ble',
      'alli'=>'al', 'entli'=>'ent', 'eli'=>'e', 'ousli'=>'ous', 'ization'=>'ize', 'ation'=>'ate',
      'ator'=>'ate', 'alism'=>'al', 'iveness'=>'ive', 'fulness'=>'ful', 'ousness'=>'ous', 'aliti'=>'al',
      'iviti'=>'ive', 'biliti'=>'ble', 'logi'=>'log');

    %step3list =
    ('icate'=>'ic', 'ative'=>'', 'alize'=>'al', 'iciti'=>'ic', 'ical'=>'ic', 'ful'=>'', 'ness'=>'');


    $c = "[^aeiou]"; # consonant
    $v = "[aeiouy]"; # vowel
    $C = "${c}[^aeiouy]*"; # consonant sequence
    $V = "${v}[aeiou]*"; # vowel sequence

    $mgr0 = "^(${C})?${V}${C}"; # [C]VC... is m>0
    $meq1 = "^(${C})?${V}${C}(${V})?" . '$'; # [C]VC[V] is m=1
    $mgr1 = "^(${C})?${V}${C}${V}${C}"; # [C]VCVC... is m>1
    $_v = "^(${C})?${v}"; # vowel in stem

}

# that's the definition. Run initialise() to set things up, then stem($word) to stem $word, as here:


# As an easy speed-up, one might create a hash of word=>stemmed form, and look up each new
# word in the hash, only calling stem() if the word was not found there.

initialise();

1;
```

Listing D.6: src/org/recursed/w/server/rss/Source.pm

```
package Source;

sub new {
 my $class = shift;
 my $self = {
  id=>shift,
  codename => shift,
  rssurl=>shift,
 };
 bless $self, $class;
 return $self;
}

1;
```

Listing D.7: src/org/recursed/w/server/rss/Word.pm

```
package Word;

sub new {
 my $class = shift;
 my $self = {
  id=>shift,
  label=>shift,
  frequencySum=>shift,
 };
 bless $self, $class;
 return $self;
}

1;
```

Listing D.8: src/org/recursed/w/server/rss/WordNetSupport.pm

```
use warnings;
use strict;
use WordNet::QueryData;

my $wn = WordNet::QueryData->new();
my @nounList = sort( $wn->listAllWords("noun") );
my @verbList = sort( $wn->listAllWords("verb") );
#print "Nouns (", scalar @nounList, ") and verbs (", scalar @verbList,
# ") loaded.";
#print "\n";
my %foundNouns = ();

sub binSearch {
 my $result = 0;
 my $searchFor = shift;
 my $cachedValue = $foundNouns{$searchFor};
 if (defined($cachedValue)) {
```

```
  #print "*";
  if ($cachedValue == 1) {
   return 1;
  } else {
   return 0;
  }
 } else {
  #print "-";
 }
 my @list = @nounList;
 my $listSize = scalar @list;
 my $left = 0;
 my $right = $listSize - 1;
 my $middle = 0;
 my $direction;
 while ( $left < $right) {
  #print "While searching for $searchFor we are at $left $middle $right\n";
  $middle = sprintf("%.0f", $left + ( ( $right - $left ) / 2 ));
  if ( $list[$middle] eq $searchFor ) {
   $result = 1;
   $foundNouns{$searchFor}=1;
   return $result;
  }
  else {
   if ( $list[$middle] lt $searchFor ) {
    $left = $middle + 1;
   }
   else {
    $right = $middle -1;
   }
  }
 }
 $foundNouns{$searchFor} = 2;
 return $result;
}

sub lookupWord {
 my $word = shift;
 if ($word eq "") {
  return 0;
 }
 if (binSearch ($word) == 1) {
  return 1;
 }
 return 0;
}

1;
```

D.3 Reporting (Website Server and Frontend)

Listing D.9: src/org/recursed/w/client/about/AboutDisplay.java

```
package org.recursed.w.client.about;

import org.recursed.w.client.characteristics.HasVisibility;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class AboutDisplay implements AboutPresenter.Display {

    private static String OVERLAY_ID = "overlay";
    private static String ABOUT_ID = "about";
    private static String CLOSE_STYLE = "close";
    private static String STYLE_OVERLAY = "blackout";
    private static String STYLE_ABOUT = "about";

    private RootPanel overlay;
    private RootPanel aboutDiv;
    private Label close;
    private HasVisibility visibility;

    @Inject
    public AboutDisplay() {
        overlay = RootPanel.get(OVERLAY_ID);
        aboutDiv = RootPanel.get(ABOUT_ID);
        close = new Label("close");
        close.setStyleName(CLOSE_STYLE);
        aboutDiv.add(close);
        createVisibility();
    }

    private void createVisibility() {
        visibility = new HasVisibility() {

            @Override
            public void setVisibility(boolean isVisibile) {
                if (isVisibile) {
                    overlay.setVisible(true);
                    aboutDiv.setVisible(true);
                } else {
                    overlay.setVisible(false);
                    aboutDiv.setVisible(false);
                }
            }
        };
        overlay.setStyleName(STYLE_OVERLAY);
        aboutDiv.setStyleName(STYLE_ABOUT);
        visibility.setVisibility(false);
    }

    @Override
    public HasClickHandlers getCloseButton() {
        return close;
    }
```

```
    @Override
    public HasVisibility getVisibility() {
        return visibility;
    }

}
```

Listing D.10: src/org/recursed/w/client/about/AboutPresenter.java

```
package org.recursed.w.client.about;

import org.recursed.w.client.characteristics.HasVisibility;
import org.recursed.w.client.place.AboutPlace;
import org.recursed.w.client.place.PlaceChangeEvent;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.SummaryPlace;
import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class AboutPresenter {

    public interface Display {
        public HasClickHandlers getCloseButton();

        public HasVisibility getVisibility();
    }

    private Display display;
    private PlaceService placeService;

    @Inject
    public AboutPresenter(Display display, PlaceService placeService) {
        this.display = display;
        this.placeService = placeService;
        initButtons();
        initPlaceService();
    }

    private void initPlaceService() {
        placeService.addPlaceChangeHandler(new PlaceChangeHandler() {

            @Override
            public void onPlaceChangeEvent(PlaceChangeEvent pce) {
                if (pce.getPlace() instanceof AboutPlace) {
                    display.getVisibility().setVisibility(true);
                }
            }
        });
```

```
    }

    private void initButtons() {
        display.getCloseButton().addClickHandler(new ClickHandler() {

            @Override
            public void onClick(ClickEvent event) {
                placeService.changePlace(new SummaryPlace());
                display.getVisibility().setVisibility(false);
            }
        });
    }
}
```

Listing D.11: src/org/recursed/w/client/characteristics/HasChart.java

```
package org.recursed.w.client.characteristics;

public interface HasChart<SERIES, X,Y> {

    public HasChartControls getControls();
    public HasSeries<X,Y> getSeries(SERIES series);
}
```

Listing D.12: src/org/recursed/w/client/characteristics/HasChartControls.java

```
package org.recursed.w.client.characteristics;

public interface HasChartControls {

    public void clear();

    public void render();
}
```

Listing D.13: src/org/recursed/w/client/characteristics/HasData.java

```
package org.recursed.w.client.characteristics;

public interface HasData<T> {

    public T getData();

    public void setData(T t);
}
```

Listing D.14: src/org/recursed/w/client/characteristics/HasDataPoint.java

```
package org.recursed.w.client.characteristics;

import com.google.gwt.user.client.ui.HasText;

public interface HasDataPoint<X,Y> {

    public HasText getLabel();
    public HasData<X> getX();
    public HasData<Y> getY();

}
```

Listing D.15: src/org/recursed/w/client/characteristics/HasHeadline.java

```
package org.recursed.w.client.characteristics;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.HasText;

public interface HasHeadline {

    public HasText getString();
    public HasClickHandlers getClickHandlers();
}
```

Listing D.16: src/org/recursed/w/client/characteristics/HasSeries.java

```
package org.recursed.w.client.characteristics;

import com.google.gwt.event.dom.client.HasClickHandlers;


public interface HasSeries<X,Y> {

    public HasDataPoint<X, Y> createDataPoint();
    public HasClickHandlers getClickHandlers();
}
```

Listing D.17: src/org/recursed/w/client/characteristics/HasVisibility.java

```
package org.recursed.w.client.characteristics;

public interface HasVisibility {

    public void setVisibility(boolean isVisibile);

}
```

Listing D.18: src/org/recursed/w/client/characteristics/HasWord.java

```
package org.recursed.w.client.characteristics;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.HasText;

public interface HasWord {

    public HasClickHandlers getClickHandlers();

    public HasText getText();

}
```

Listing D.19: src/org/recursed/w/client/guice/ActionServiceProvider.java

```
package org.recursed.w.client.guice;

import org.recursed.w.shared.corerpc.ActionService;
import org.recursed.w.shared.corerpc.ActionServiceAsync;

import com.google.gwt.core.client.GWT;
import com.google.inject.Provider;
import com.google.inject.Singleton;

@Singleton public class ActionServiceProvider implements Provider<ActionServiceAsync>{

    private ActionServiceAsync service;

    public ActionServiceProvider() {
        service = GWT.create(ActionService.class);
    }

    @Override
    public ActionServiceAsync get() {
        return service;
    }
}
```

Listing D.20: src/org/recursed/w/client/guice/RPCServiceProvider.java

```
package org.recursed.w.client.guice;

import org.recursed.w.client.rpc.BatchingRPCService;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.client.rpc.RootRPCServiceImpl;
import org.recursed.w.shared.corerpc.ActionServiceAsync;

import com.google.inject.Inject;
import com.google.inject.Provider;
import com.google.inject.Singleton;
```

```
@Singleton
public class RPCServiceProvider implements Provider<RPCService>{

    private RPCService rpcService;

    @Inject
    public RPCServiceProvider(ActionServiceAsync actionServiceAsync) {
        RootRPCServiceImpl root = new RootRPCServiceImpl(actionServiceAsync);
        rpcService = new BatchingRPCService(root);
    }

    @Override
    public RPCService get() {
        return rpcService;
    }

}
```

Listing D.21: src/org/recursed/w/client/guice/WClientModule.java

```
package org.recursed.w.client.guice;

import org.recursed.w.client.about.AboutDisplay;
import org.recursed.w.client.about.AboutPresenter;
import org.recursed.w.client.headlines.HeadlinesDisplay;
import org.recursed.w.client.headlines.HeadlinesPresenter;
import org.recursed.w.client.place.PlaceFactory;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.relationshipgraph.RelationshipGraphDisplay;
import org.recursed.w.client.relationshipgraph.RelationshipGraphPresenter;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.client.search.SearchDisplay;
import org.recursed.w.client.search.SearchPresenter;
import org.recursed.w.client.search.WordSuggestOracle;
import org.recursed.w.client.summary.SummaryDisplay;
import org.recursed.w.client.summary.SummaryPresenter;
import org.recursed.w.client.worddetail.ChosenWordDisplay;
import org.recursed.w.client.worddetail.ChosenWordPresenter;
import org.recursed.w.shared.corerpc.ActionServiceAsync;

import com.google.gwt.inject.client.AbstractGinModule;
import com.google.inject.Singleton;

public class WClientModule extends AbstractGinModule {

    @Override
    protected void configure() {
        bind(ActionServiceAsync.class).toProvider(ActionServiceProvider.class);
        bind(RPCService.class).toProvider(RPCServiceProvider.class);
        bind(PlaceService.Factory.class).to(PlaceFactory.class).in(Singleton.class);
        bind(PlaceService.class).in(Singleton.class);
        bind(SummaryPresenter.Display.class).to(SummaryDisplay.class).in(Singleton.class);
        bind(ChosenWordPresenter.Display.class).to(ChosenWordDisplay.class).in(Singleton.class);
        bind(ChosenWordPresenter.class).in(Singleton.class);
```

```
        bind(RelationshipGraphPresenter.Display.class).to(RelationshipGraphDisplay.class).in(
            Singleton.class);
        bind(RelationshipGraphPresenter.class).in(Singleton.class);
        bind(AboutPresenter.Display.class).to(AboutDisplay.class).in(Singleton.class);
        bind(AboutPresenter.class).in(Singleton.class);
        bind(HeadlinesPresenter.Display.class).to(HeadlinesDisplay.class).in(Singleton.class);
        bind(HeadlinesPresenter.class).in(Singleton.class);
        bind(WordSuggestOracle.class).in(Singleton.class);
        bind(SearchPresenter.Display.class).to(SearchDisplay.class).in(Singleton.class);
        bind(SearchPresenter.class).in(Singleton.class);
    }

}
```

Listing D.22: src/org/recursed/w/client/guice/WGinjector.java

```
package org.recursed.w.client.guice;

import org.recursed.w.client.about.AboutPresenter;
import org.recursed.w.client.headlines.HeadlinesPresenter;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.client.search.SearchPresenter;
import org.recursed.w.client.summary.SummaryPresenter;
import org.recursed.w.client.worddetail.ChosenWordPresenter;

import com.google.gwt.inject.client.GinModules;
import com.google.gwt.inject.client.Ginjector;

@GinModules(WClientModule.class)
public interface WGinjector extends Ginjector {
    public RPCService getRPCService();
    public PlaceService getPlaceService();
    public SummaryPresenter getSummaryPresenter();
    public ChosenWordPresenter getChosenWordPresenter();
    public AboutPresenter getAboutPresenter();
    public HeadlinesPresenter getHeadlinesPresenter();
    public SearchPresenter getSearchPresenter();
}
```

Listing D.23: src/org/recursed/w/client/headlines/HeadlinesDisplay.java

```
package org.recursed.w.client.headlines;

import org.recursed.w.client.characteristics.HasHeadline;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.HasText;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.inject.Inject;
import com.google.inject.Singleton;
```

```
@Singleton
public class HeadlinesDisplay implements HeadlinesPresenter.Display {

    private static final String HEADLINES_DIV_ID = "headlines";
    private static final String HEADLINE_STYLE = "headline";
    private HasHeadline[] headlines;

    @Inject
    public HeadlinesDisplay() {
        RootPanel myDiv = RootPanel.get(HEADLINES_DIV_ID);
        headlines = new HasHeadline[MAX_HEADLINES];
        for (int x=0; x<MAX_HEADLINES; x++) {
            final Label l = new Label();
            l.setStyleName(HEADLINE_STYLE);
            headlines[x] = new HasHeadline() {

                @Override
                public HasText getString() {
                    return l;
                }

                @Override
                public HasClickHandlers getClickHandlers() {
                    return l;
                }
            };
            myDiv.add(l);
        }
    }

    @Override
    public HasHeadline getHeadline(int position) {
        return headlines[position];
    }

}
```

Listing D.24: src/org/recursed/w/client/headlines/HeadlinesPresenter.java

```
package org.recursed.w.client.headlines;

import java.util.ArrayList;

import org.recursed.w.client.characteristics.HasHeadline;
import org.recursed.w.client.place.PlaceChangeEvent;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.shared.headlines.GetHeadlines;
import org.recursed.w.shared.headlines.GetHeadlinesCallback;
import org.recursed.w.shared.headlines.HeadlineDTO;

import com.google.gwt.event.dom.client.ClickEvent;
```

```
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class HeadlinesPresenter {

    private RPCService rpcService;
    private GetHeadlinesCallback callback;
    private ArrayList<HeadlineDTO> currentHeadlines;
    private int counter;
    private HasHeadline[] hasHeadlines;

    public interface Display {
        public static final int MAX_HEADLINES = 15;
        public HasHeadline getHeadline(int position);
    }

    @Inject
    public HeadlinesPresenter(RPCService rpcService, PlaceService placeService, Display display
        ) {
        this.rpcService = rpcService;
        placeService.addPlaceChangeHandler(new PlaceChangeHandler() {

            @Override
            public void onPlaceChangeEvent(PlaceChangeEvent pce) {
                if (pce.getPlace() instanceof WordDetailPlace) {
                    WordDetailPlace wordDetailPlace = (WordDetailPlace) pce.getPlace();
                    int wordId = wordDetailPlace.getWordId();
                    populateHeadlinesForWord(wordId);
                }
            }
        });
        callback = new GetHeadlinesCallback() {

            @Override
            public void gotHeadlines(ArrayList<HeadlineDTO> headlines) {
                currentHeadlines = headlines;
                counter = 0;
                repopulate();
            }
        };
        hasHeadlines = new HasHeadline[Display.MAX_HEADLINES];
        for (int x=0; x<Display.MAX_HEADLINES; x++) {
            final int position = x;
            hasHeadlines[x] = display.getHeadline(x);
            hasHeadlines[x].getClickHandlers().addClickHandler(new ClickHandler() {

                @Override
                public void onClick(ClickEvent event) {
                    showLink(position);
                }
            });
        }
    }
```

```
    private void populateHeadlinesForWord(int wordId) {
        rpcService.execute(new GetHeadlines(wordId), callback);
    }

    private void repopulate() {
        for (int x=0; x<Display.MAX_HEADLINES; x++) {
            hasHeadlines[x].getString().setText(null);
        }
        for (int x=0; x<Display.MAX_HEADLINES && x+counter < currentHeadlines.size(); x++) {
            HeadlineDTO dto = currentHeadlines.get(x+counter);
            hasHeadlines[x].getString().setText(dto.getTitle());
        }
    }

    public void showLink(int position) {
        HeadlineDTO toShow = currentHeadlines.get(position + counter);
        String link = toShow.getLink();
        Window.open(link, "_blank", null);
    }
}
```

Listing D.25: src/org/recursed/w/client/place/AboutPlace.java

```
package org.recursed.w.client.place;

public class AboutPlace extends Place {

    public static final String SHORT_CODE = "a";
    public AboutPlace(String urlShortCode) {
        super(SHORT_CODE);
    }

    public AboutPlace() {
        super(SHORT_CODE);
    }

    @Override
    public String[] getTokenizedArguments() {
        return new String[] {};
    }

    @Override
    public void setTokens(String[] tokens) {
        // do nothing
    }

}
```

Listing D.26: src/org/recursed/w/client/place/Place.java

```
package org.recursed.w.client.place;
```

```
import com.google.gwt.user.client.Command;

public abstract class Place {

    private String urlShortCode;
    /**
     * Mandatory..
     * @param urlShortCode
     */
    public Place(String urlShortCode) {
        this.urlShortCode = urlShortCode;
    }

    /**
     * @return A representation for this place's name on the url.
     * This should be something like 's' for 'SomePlace'
     */
    public final String getURLShortCode() {
        return urlShortCode;
    }

    /**
     * @return Optional arguments for the place based on the DTOs
     * within it. Eg: SomePlace may contain a Person with id 1, so
     * it would return something like {"p", "1"}
     */
    public abstract String[] getTokenizedArguments();

    /**
     * This method is fired on a brower driven place change (or
     * when a url is changed). The supplied tokens are what this place
     * would have returned via the tokenizeArguments method. This
     * object should take the opportunity to flesh out any models/dtos
     * that it may normally provide. When done, it should fire the
     * supplied command.
     *
     * @param tokens
     * @param c
     */
    public void inflateAndFire(Command c) {
        c.execute();
    }

    public abstract void setTokens(String[] tokens);

}
```

Listing D.27: src/org/recursed/w/client/place/PlaceChangeEvent.java

```
package org.recursed.w.client.place;

import com.google.gwt.event.shared.EventHandler;
import com.google.gwt.event.shared.GwtEvent;

public class PlaceChangeEvent extends
```

```
        GwtEvent<PlaceChangeEvent.PlaceChangeHandler> {

    public interface PlaceChangeHandler extends EventHandler {
        public void onPlaceChangeEvent(PlaceChangeEvent pce);

    }

    public static final GwtEvent.Type<PlaceChangeHandler> TYPE = new GwtEvent.Type<
         PlaceChangeHandler>();

    private Place p;

    public PlaceChangeEvent(Place p) {
        this.p = p;
    }

    public Place getPlace() {
        return p;
    }

    @Override
    public com.google.gwt.event.shared.GwtEvent.Type<PlaceChangeHandler> getAssociatedType() {
        return TYPE;
    }

    @Override
    protected void dispatch(PlaceChangeHandler handler) {
        handler.onPlaceChangeEvent(this);
    }

}
```

Listing D.28: src/org/recursed/w/client/place/PlaceFactory.java

```
package org.recursed.w.client.place;

import java.util.HashMap;
import java.util.Map;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class PlaceFactory implements PlaceService.Factory {

    private Map<String, Class<? extends Place>> places;

    @Inject
    private PlaceFactory() {
        places = new HashMap<String, Class<? extends Place>>();
        places.put(SummaryPlace.SHORT_CODE, SummaryPlace.class);
        places.put(WordDetailPlace.CODE, WordDetailPlace.class);
        places.put(AboutPlace.SHORT_CODE, AboutPlace.class);
    }
```

```
    @Override
    public Place getDefaultPlace() {
        return new SummaryPlace();
    }

    @Override
    public Place getPlace(String urlShortCode, String[] tokens) {
        Place p = null;
        if (urlShortCode.equals("s")) {
            p = new SummaryPlace();
        } else if (urlShortCode.equals("w")) {
            p = new WordDetailPlace();
        } else if (urlShortCode.equals("a")) {
            p = new AboutPlace();
        }
        if (p == null) {
            p = getDefaultPlace();
        }
        p.setTokens(tokens);
        return p;
    }
}
```

Listing D.29: src/org/recursed/w/client/place/PlaceService.java

```
package org.recursed.w.client.place;

import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.event.shared.HandlerManager;
import com.google.gwt.event.shared.HandlerRegistration;
import com.google.gwt.user.client.Command;
import com.google.gwt.user.client.History;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class PlaceService implements ValueChangeHandler<String> {

    public interface Factory {
        /**
         * Given the current arguments on the history, resolve a Place. Note
         * that the arguments will be non-null or greater than zero.
         *
         * @param urlShortCode
         * @return
         */
        public Place getPlace(String urlShortCode, String[] tokens);

        /**
         * When the application first fires up, or when a zero-argument place
         * change is detected, the Factory will be asked to provide a Place this
```

```
     * service will change to.
     *
     * @return
     */
    public Place getDefaultPlace();

}

private Factory factory;
private HandlerManager handler;

@Inject
private PlaceService(Factory factory) {
    this.factory = factory;
    this.handler = new HandlerManager(null);
    History.addValueChangeHandler(this);
}

public void fireInitialPlace() {
    String initialToken = History.getToken();
    if (initialToken == null || initialToken.length() == 0) {
        Place start = factory.getDefaultPlace();
        updateHistoryWithPlace(start);
    } else {
        processFragment(initialToken);
    }
}

public HandlerRegistration addPlaceChangeHandler(PlaceChangeHandler handler) {
    return this.handler.addHandler(PlaceChangeEvent.TYPE, handler);
}

public void changePlace(Place p) {
    updateHistoryWithPlace(p);
}

private String getStringForPlace(Place p) {
    String placeShortCode = p.getURLShortCode();
    StringBuilder result = new StringBuilder(16);
    result.append(placeShortCode);
    String[] tokens = p.getTokenizedArguments();
    if (tokens != null && tokens.length > 0) {
        for (String token : tokens) {
            result.append(':');
            result.append(token);
        }
    }
    return result.toString();
}

private void updateHistoryWithPlace(Place p) {
    String newHistoryItem = getStringForPlace(p);
    History.newItem(newHistoryItem, false);
    PlaceChangeEvent e = new PlaceChangeEvent(p);
    handler.fireEvent(e);
    registerPlaceInGoogleAnalytics(newHistoryItem);
}
```

```
    @Override
    public void onValueChange(ValueChangeEvent<String> event) {
        String fullString = event.getValue();
        processFragment(fullString);
        registerPlaceInGoogleAnalytics(fullString);
    }

    private void processFragment(String fullString) {
        int firstColon = fullString.indexOf(':');
        Place p = null;
        if (firstColon > 0) {
            // we have arguments..
            String urlShortCode = fullString.substring(0, firstColon);
            String remainingArgs = fullString.substring(firstColon);
            String[] arguments = remainingArgs.split(":");
            p = factory.getPlace(urlShortCode, arguments);
        } else {
            // no arguments
            p = factory.getPlace(fullString, new String[] {});
        }
        final Place toFire = p;
        Command c = new Command() {

            @Override
            public void execute() {
                handler.fireEvent(new PlaceChangeEvent(toFire));
            }

        };
        p.inflateAndFire(c);
    }

    private void registerPlaceInGoogleAnalytics(String fragment) {
        try {
            registerPlaceInGoogleAnalyticsNative('/'+fragment);
        } catch (Exception e) {
            GWT.log("Failed to talk to GA", e);
        }
    }
    private native void registerPlaceInGoogleAnalyticsNative(String fragment) /*-{
        $wnd._gaq.push(['_trackPageview', fragment]);
    }-*/;

}
```

Listing D.30: src/org/recursed/w/client/place/SummaryPlace.java

```
package org.recursed.w.client.place;


public class SummaryPlace extends Place {

    public static final String SHORT_CODE = "s";
```

```
    public SummaryPlace() {
        super(SHORT_CODE);
    }

    @Override
    public void setTokens(String[] tokens) {
        // do nothing..
    }

    @Override
    public String[] getTokenizedArguments() {
        return new String[]{};
    }

}
```

Listing D.31: src/org/recursed/w/client/place/WordDetailPlace.java

```
package org.recursed.w.client.place;

import org.recursed.w.shared.WordDTO;

public class WordDetailPlace extends Place {

    public static final String CODE = "w";
    /**
     * This is there for programmatic change places to decorate the URL.
     * Do not assume that it will be there for all place changes (and therefore expose it)
     * since browser driven ones may not have a reliable WordDTO. Bank on the ID only.
     */
    private WordDTO dto;
    private int id;

    public WordDetailPlace() {
        super(CODE);
    }

    public WordDetailPlace(WordDTO dto) {
        super(CODE);
        this.id = dto.getId();
        this.dto = dto;
    }


    public int getWordId() {
        return id;
    }

    @Override
    public String[] getTokenizedArguments() {
        if (this.dto != null) {
            return new String[] { "" + id, dto.getDisplayString() };
        } else {
            return new String[] { "" + id };
        }
```

```
    }

    @Override
    public void setTokens(String[] tokens) {
        if (tokens.length > 1) {
            id = new Integer(tokens[1]);
        }
    }

}
```

Listing D.32: src/org/recursed/w/client/relationshipgraph/GetGroupedWordDetailCallback.java

```
package org.recursed.w.client.relationshipgraph;

import org.recursed.w.client.rpc.ActionCallback;
import org.recursed.w.shared.GroupedWordDetail;

public abstract class GetGroupedWordDetailCallback extends ActionCallback<GroupedWordDetail
    > {

}
```

Listing D.33: src/org/recursed/w/client/relationshipgraph/RelationshipGraphDisplay.java

```
package org.recursed.w.client.relationshipgraph;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;

import org.recursed.w.client.characteristics.HasChartControls;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.shared.WordDTO;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.shared.HandlerRegistration;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.widgetideas.graphics.client.Color;
import com.google.gwt.widgetideas.graphics.client.GWTCanvas;
import com.google.inject.Inject;

public class RelationshipGraphDisplay implements
        RelationshipGraphPresenter.Display {

    private static final String GRAPH_DIV = "graph";
```

```
private static final String WORD_TITLE_STYLE = "word_title_style";
private static final String LABEL_STYLE = "middle";
private static final Color DK_GRAY = new Color("#BB2222");
private static final Color[] SECONDARY = new Color[] {
    new Color("#553C31"),
    new Color("#EE5313"),
    new Color("#7f7f7f"),
    new Color("#54C354"),
    new Color("#60EE13"),
    new Color("#CA52F6")
};

private int width;
private int height;
private double middleX, middleY;

private PlaceService placeService;
private Label wordTitle;
private GWTCanvas canvas;
private AbsolutePanel absolutePanel;
private HasChartControls controls;
private ArrayList<double[]> phaseOneCoords;
private ArrayList<double[]> phaseTwoCoords;
private HashSet<WordDTO> secondaryWords;
private ArrayList<WordDTO[]> secondaryTuples;
private HashMap<WordDTO, Color> colors;
private HashMap<Integer, double[]> coordinates;
private int currentPhaseOneLinkage, currentPhaseTwoLinkage;
private ArrayList<Label> labels = new ArrayList<Label>();
private ArrayList<HandlerRegistration> registrations = new ArrayList<
    HandlerRegistration>();

@Inject
private RelationshipGraphDisplay(PlaceService ps) {
    width = 480;
    height = 480;
    this.placeService = ps;
    absolutePanel = new AbsolutePanel();
    canvas = new GWTCanvas(width, height);
    absolutePanel.add(canvas);
    RootPanel graphPanel = RootPanel.get(GRAPH_DIV);
    wordTitle = new Label();
    wordTitle.setStyleName(WORD_TITLE_STYLE);
    graphPanel.add(wordTitle);
    graphPanel.add(absolutePanel);
    createChartControls();
}

private void createChartControls() {
    controls = new HasChartControls() {

        @Override
        public void render() {
            // does nothing..
        }

        @Override
```

```
        public void clear() {
            canvas.clear();
            for (Label l : labels) {
                absolutePanel.remove(l);
            }
            labels = new ArrayList<Label>();
            for (HandlerRegistration r : registrations) {
                r.removeHandler();
            }
            registrations = new ArrayList<HandlerRegistration>();
        }
    };
}

@Override
public HasChartControls getControls() {
    return controls;
}

@Override
public void setCoreWord(WordDTO wordDTO) {
    middleX = width/2;
    middleY = height/2;
    coordinates = new HashMap<Integer, double[]>();
    phaseOneCoords = new ArrayList<double[]>();
    phaseTwoCoords = new ArrayList<double[]>();
    secondaryTuples = new ArrayList<WordDTO[]>();
    secondaryWords = new HashSet<WordDTO>();
    colors = new HashMap<WordDTO, Color>();
    computeCoords(phaseOneCoords, 100d, 6, 0);
    currentPhaseOneLinkage = 0;
    currentPhaseTwoLinkage = 0;
    drawWord(new double[] {middleX, middleY}, 10, wordDTO, DK_GRAY);
    wordTitle.setText(wordDTO.getDisplayString());
}

private void computeCoords(ArrayList<double[]> collection, double radius, int nodes,
     double offset) {
    double degreeSegments = (2d*Math.PI)/nodes;
    for (int x=0; x < nodes; x++) {
        double newX = radius * Math.cos(offset + (degreeSegments * x));
        double newY = radius * Math.sin(offset + (degreeSegments * x));
        collection.add(new double[]{middleX + newX, middleY + newY});
    }
}

@Override
public void addSecondaryLinkage(WordDTO from, WordDTO to) {
    if (coordinates.get(to.getId()) == null) {
        // this is a new word, not in the inner ring
        secondaryWords.add(to);
    }
    secondaryTuples.add(new WordDTO[] {from, to});
}

@Override
public void renderSecondaryLinkages() {
```

```
        computeCoords(phaseTwoCoords, 200d, secondaryWords.size(), 0.5);
        for (WordDTO toWord : secondaryWords) {
            double[] coords = phaseTwoCoords.get(currentPhaseTwoLinkage);
            drawWord(coords, 3, toWord, null);
            currentPhaseTwoLinkage++;
        }
        canvas.setLineWidth(1);
        for (WordDTO[] tuple : secondaryTuples) {
            Color c = colors.get(tuple[0]);
            canvas.setStrokeStyle(c);
            drawLinkage(tuple[0], tuple[1]);
        }
    }

    @Override
    public void addLinkage(WordDTO from, WordDTO to) {
        if (currentPhaseOneLinkage > phaseOneCoords.size()) {
            throw new IllegalArgumentException("Exceeded linkage count");
        }
        double[] coords = phaseOneCoords.get(currentPhaseOneLinkage);
        currentPhaseOneLinkage++;
        canvas.setStrokeStyle(DK_GRAY);
        canvas.setLineWidth(2);
        Color c = colors.get(to);
        if (c == null) {
            c = SECONDARY[colors.size()];
            colors.put(to, c);
        }
        drawWord(coords, 3, to, c);
        drawLinkage(from, to);
    }

    private void drawLinkage(WordDTO from, WordDTO to) {
        double[] fc = coordinates.get(from.getId());
        double[] tc = coordinates.get(to.getId());
        canvas.beginPath();
        canvas.moveTo(fc[0], fc[1]);
        //canvas.lineTo(tc[0], tc[1]);
        canvas.quadraticCurveTo(tc[0]-15, tc[1]-15, tc[0], tc[1]);
        canvas.stroke();
    }

    private void drawWord(double[] coords, double radius, final WordDTO dto, Color c) {
        int textLength = dto.getDisplayString().length();
        Label l = new Label(dto.getDisplayString());
        l.setStylePrimaryName(LABEL_STYLE);
        int x = new Double(coords[0]).intValue()-(6*(textLength/2));
        int y = new Double(coords[1]).intValue()-5;
        absolutePanel.add(l, x, y);
        coordinates.put(dto.getId(), coords);
        labels.add(l);
        registrations.add(l.addClickHandler(new ClickHandler() {

            @Override
            public void onClick(ClickEvent event) {
                placeService.changePlace(new WordDetailPlace(dto));
            }
```

```
        }));
        if (c != null) {
            // need to draw a box
            canvas.beginPath();
            canvas.setFillStyle(c);
            double width = l.getOffsetWidth();
            double height = l.getOffsetHeight();
            canvas.fillRect(x-2, y-2, width+4, height+4);
        }
    }

}
```

Listing D.34: src/org/recursed/w/client/relationshipgraph/RelationshipGraphPresenter.java

```
package org.recursed.w.client.relationshipgraph;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

import org.recursed.w.client.characteristics.HasChartControls;
import org.recursed.w.client.place.Place;
import org.recursed.w.client.place.PlaceChangeEvent;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.client.worddetail.GetWordDetailCallback;
import org.recursed.w.shared.GetGroupedWordDetailAction;
import org.recursed.w.shared.GetWordDetailAction;
import org.recursed.w.shared.GroupedWordDetail;
import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.WordDetail;

import com.google.inject.Inject;

public class RelationshipGraphPresenter implements PlaceChangeHandler {

    private WordDTO core;

    public interface Display {
        public HasChartControls getControls();

        public void setCoreWord(WordDTO wordDTO);

        public void addLinkage(WordDTO from, WordDTO to);

        public void addSecondaryLinkage(WordDTO from, WordDTO to);

        public void renderSecondaryLinkages();
    }
```

```
private Display display;
private RPCService rpcService;
private PlaceService placeService;
private int expectingDataFor;

@Inject
public RelationshipGraphPresenter(Display display,
        RPCService rpcService, PlaceService placeService) {
    this.display = display;
    this.rpcService = rpcService;
    this.placeService = placeService;
    this.placeService.addPlaceChangeHandler(this);
}

private void reset(int id) {
    GetWordDetailAction gwa = new GetWordDetailAction(id, GetWordDetailAction.
         DEFAULT_MAX_WORDS);
    rpcService.execute(gwa, new GetWordDetailCallback() {

        @Override
        public void got(WordDetail response) {
            WordDTO coreWord = response.getCoreWord();
            int gotId = coreWord.getId();
            if (gotId != expectingDataFor) {
                return;
            }
            resetGraph(response);
        }
    });
}

private void resetGraph(WordDetail result) {
    core = result.getCoreWord();
    display.getControls().clear();
    display.setCoreWord(core);
    HashSet<WordDTO> relatedWords = result.getRandomRelatedWords(6);
    GetGroupedWordDetailAction moreWords = new GetGroupedWordDetailAction();
    for (WordDTO relatedWord : relatedWords) {
        display.addLinkage(core, relatedWord);
        moreWords.addWord(relatedWord);
    }
    rpcService.execute(moreWords, new GetGroupedWordDetailCallback() {

        @Override
        public void got(GroupedWordDetail result) {
            HashMap<WordDTO,ArrayList<WordDTO>> linkages = result.getLinkages();
            Set<WordDTO> froms = linkages.keySet();
            for (WordDTO from : froms) {
                ArrayList<WordDTO> tos = linkages.get(from);
                for (WordDTO to : tos) {
                    display.addSecondaryLinkage(from, to);
                    //GWT.log("Adding " + from.getLabel() + " to " + to.getLabel(), null);
                }
            }
            display.renderSecondaryLinkages();
        }
```

```
        });
    }

    @Override
    public void onPlaceChangeEvent(PlaceChangeEvent pce) {
        Place place = pce.getPlace();
        if (!(place instanceof WordDetailPlace)) {
            return;
        }
        WordDetailPlace wdp = (WordDetailPlace) place;
        int id = wdp.getWordId();
        expectingDataFor = id;
        reset(id);
    }

}
```

Listing D.35: src/org/recursed/w/client/rpc/ActionCallback.java

```
package org.recursed.w.client.rpc;

import org.recursed.w.shared.corerpc.Response;

import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;

public abstract class ActionCallback<R extends Response> implements AsyncCallback<Response
    >{

    /**
     * Centrally managed failures..
     */
    @Override
    public void onFailure(Throwable caught) {
        RootPanel.get().add(new Label(caught.toString()));
    }

    @SuppressWarnings("unchecked")
    @Override
    public final void onSuccess(Response result) {
        got((R) result);
    }

    public abstract void got(R response);
}
```

Listing D.36: src/org/recursed/w/client/rpc/BatchedActionCallback.java

```
package org.recursed.w.client.rpc;

import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.Set;

import org.recursed.w.shared.corerpc.BatchedResponse;
import org.recursed.w.shared.corerpc.Response;

public class BatchedActionCallback extends ActionCallback<BatchedResponse>{

    private HashMap<Integer, ArrayList<ActionCallback<?>>> callbackMap = new HashMap<
         Integer, ArrayList<ActionCallback<?>>>();

    public void addCallback(Integer counter, ActionCallback<?> callback) {
        ArrayList<ActionCallback<?>> callbacks = callbackMap.get(counter);
        if (callbacks == null) {
            callbacks = new ArrayList<ActionCallback<?>>();
        }
        callbacks.add(callback);
        callbackMap.put(counter, callbacks);
    }

    @Override
    public void got(BatchedResponse result) {
        Set<Integer> keySet = callbackMap.keySet();
        for (Integer key : keySet) {
            ArrayList<ActionCallback<?>> callbacks = callbackMap.get(key);
            Response r = result.getResponse(key);
            for (ActionCallback<?> callback : callbacks) {
                if (r != null) {
                    callback.onSuccess(r);
                } else {
                    Exception e = result.getException(key);
                    callback.onFailure(e);
                }
            }
        }
    }

}
```

Listing D.37: src/org/recursed/w/client/rpc/BatchingRPCService.java

```
package org.recursed.w.client.rpc;

import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.BatchedAction;
import org.recursed.w.shared.corerpc.Response;

import com.google.gwt.user.client.Timer;
import com.google.inject.Singleton;

@Singleton
public class BatchingRPCService implements RPCService {

    private RPCService impl;
    private Timer t;
```

```
    private BatchedAction currentBatch;
    private BatchedActionCallback currentCallback;
    private int counter;
    private int batchCounter;
    private boolean freshBatch;

    public BatchingRPCService(RPCService impl) {
        this.impl = impl;
        t = new Timer() {

            @Override
            public void run() {
                fireBatch();
            }
        };
        freshBatch = true;
    }

    @Override
    public <A extends Action, R extends Response> void execute(A action,
            ActionCallback<R> callback) {
        //GWT.log("Batch " + batchCounter + " -> " + action.getClass(), null);
        if (freshBatch) {
            currentBatch = new BatchedAction();
            currentCallback = new BatchedActionCallback();
            counter = 0;
            t.schedule(30);
            freshBatch = false;
        }
        int actionID = currentBatch.addAction(counter, action);
        currentCallback.addCallback(actionID, callback);
        counter++;
    }

    private void fireBatch() {
        impl.execute(currentBatch, currentCallback);
        freshBatch = true;
        batchCounter ++;
    }

}
```

Listing D.38: src/org/recursed/w/client/rpc/RootRPCServiceImpl.java

```
package org.recursed.w.client.rpc;

import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.ActionServiceAsync;
import org.recursed.w.shared.corerpc.Response;

import com.google.inject.Inject;
import com.google.inject.Singleton;

/**
 * The innermost Russian doll in the rpc service pipeline.
```

```
 * @author keerat
 *
 */
@Singleton
public class RootRPCServiceImpl implements RPCService{

    private ActionServiceAsync actionService;

    @Inject
    public RootRPCServiceImpl(ActionServiceAsync actionService) {
        this.actionService = actionService;
    }

    @Override
    public<A extends Action, R extends Response> void execute(A action, ActionCallback<R>
         callback) {
        try {
            actionService.execute(action, callback);
        } catch (Exception e) {
            callback.onFailure(e);
        }
    }

}
```

Listing D.39: src/org/recursed/w/client/rpc/RPCService.java

```
package org.recursed.w.client.rpc;

import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.Response;

/**
 * This is the companion interface to the ActionService.
 *
 * @author keerat
 *
 */
public interface RPCService {
    public<A extends Action, R extends Response> void execute(A action, ActionCallback<R>
         callback);
}
```

Listing D.40: src/org/recursed/w/client/search/SearchDisplay.java

```
package org.recursed.w.client.search;

import com.google.gwt.event.dom.client.FocusEvent;
import com.google.gwt.event.dom.client.FocusHandler;
import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.SuggestBox;
```

```
import com.google.gwt.user.client.ui.SuggestOracle.Suggestion;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class SearchDisplay implements SearchPresenter.Display {
    private static final String SEARCH_CAPSULE_DIV = "search_capsule";
    private static final String STYLE_SUGGEST_BOX = "search_input";
    private static final String STYLE_POPUP = "search_popup";

    private SuggestBox wordSearchSuggestBox;

    @Inject
    public SearchDisplay(WordSuggestOracle suggestOracle) {
        RootPanel searchCapsuleDiv = RootPanel.get(SEARCH_CAPSULE_DIV);
        wordSearchSuggestBox = new SuggestBox(suggestOracle);
        wordSearchSuggestBox.setText("click here to search");
        wordSearchSuggestBox.setStyleName(STYLE_SUGGEST_BOX);
        wordSearchSuggestBox.setPopupStyleName(STYLE_POPUP);
        wordSearchSuggestBox.getTextBox().addFocusHandler(new FocusHandler() {

            @Override
            public void onFocus(FocusEvent event) {
                wordSearchSuggestBox.setText("");
            }
        });
        searchCapsuleDiv.add(wordSearchSuggestBox);
    }

    public HasSelectionHandlers<Suggestion> getSelectionHandlers() {
         return wordSearchSuggestBox;
    }

    @Override
    public int getTabIndex() {
        return wordSearchSuggestBox.getTabIndex();
    }

    @Override
    public void setAccessKey(char key) {
        wordSearchSuggestBox.setAccessKey(key);
    }

    @Override
    public void setFocus(boolean focused) {
        wordSearchSuggestBox.setFocus(focused);
    }

    @Override
    public void setTabIndex(int index) {
        wordSearchSuggestBox.setTabIndex(index);
    }

}
```

Listing D.41: src/org/recursed/w/client/search/SearchPresenter.java

```
package org.recursed.w.client.search;

import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.shared.WordDTO;

import com.google.gwt.event.logical.shared.HasSelectionHandlers;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.user.client.Timer;
import com.google.gwt.user.client.ui.Focusable;
import com.google.gwt.user.client.ui.SuggestOracle.Suggestion;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class SearchPresenter {

    public interface Display extends Focusable {
        public HasSelectionHandlers<Suggestion> getSelectionHandlers();
    }

    private Display display;
    private PlaceService placeService;

    @Inject
    public SearchPresenter(Display display, PlaceService placeService) {
        this.display = display;
        this.placeService = placeService;
        Timer t = new Timer() {

            @Override
            public void run() {
                init();
            }

        };
        t.schedule(10);
    }

    private void init() {
        this.display.getSelectionHandlers().addSelectionHandler( new SelectionHandler<
             Suggestion>() {

            @Override
            public void onSelection(SelectionEvent<Suggestion> event) {
                WordDTO word = (WordDTO) event.getSelectedItem();
                WordDetailPlace wordDetailPlace = new WordDetailPlace(word);
                placeService.changePlace(wordDetailPlace);
                display.setFocus(false);
            }
        });
    }

}
```

Listing D.42: src/org/recursed/w/client/search/WordSuggestOracle.java

```
package org.recursed.w.client.search;

import java.util.ArrayList;

import org.recursed.w.client.rpc.ActionCallback;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.search.GetSuggestedWordsAction;
import org.recursed.w.shared.search.SuggestedWordsResponse;

import com.google.gwt.user.client.ui.SuggestOracle;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class WordSuggestOracle extends SuggestOracle {

    private RPCService rpcService;

    @Inject
    public WordSuggestOracle (RPCService rpcService) {
        this.rpcService = rpcService;
    }

    @Override
    public void requestSuggestions(Request request, Callback suggestCallback) {
        GetSuggestedWordsAction action = new GetSuggestedWordsAction(request.getQuery());
        SuggestionRPCCallback callback = new SuggestionRPCCallback(request, suggestCallback);
        rpcService.execute(action, callback);
    }

    private class SuggestionRPCCallback extends ActionCallback<SuggestedWordsResponse> {

        private Request uiRequest;
        private Callback uiCallback;

        SuggestionRPCCallback(Request uiRequest, Callback uiCallback) {
            this.uiRequest = uiRequest;
            this.uiCallback = uiCallback;
        }

        @Override
        public void got(SuggestedWordsResponse response) {
            ArrayList<WordDTO> words = response.getWords();
            Response uiResponse = new Response();
            uiResponse.setSuggestions(words);
            uiCallback.onSuggestionsReady(uiRequest, uiResponse);
        }
    }


}
```

Listing D.43: src/org/recursed/w/client/summary/GetTopWordsInChronoBandsCallback.java

```
package org.recursed.w.client.summary;

import org.recursed.w.client.rpc.ActionCallback;
import org.recursed.w.shared.TopWordsInChronoBands;

public abstract class GetTopWordsInChronoBandsCallback extends ActionCallback<
    TopWordsInChronoBands>{

}
```

Listing D.44: src/org/recursed/w/client/summary/SummaryDisplay.java

```
package org.recursed.w.client.summary;

import java.util.HashMap;
import java.util.Map;

import org.recursed.w.client.characteristics.HasWord;
import org.recursed.w.shared.ChronoBands;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.Event;
import com.google.gwt.user.client.ui.HTMLPanel;
import com.google.gwt.user.client.ui.HasText;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class SummaryDisplay implements SummaryPresenter.Display {

    private static final String TRENDS = "trends";
    private static final String TREND_WORD = "trend_word_label";
    private static final String TRENDS_UL_DIV = "trends_div";
    private HTMLPanel mainArea;
    private Map<String, HasWord> hasWords;

    @Inject
    private SummaryDisplay() {
        final RootPanel trendsDiv = RootPanel.get(TRENDS);
        mainArea = new HTMLPanel(TRENDS_LIST);
        mainArea.setStyleName(TRENDS_UL_DIV);
        trendsDiv.add(mainArea);
        mainArea.sinkEvents(Event.FOCUSEVENTS | Event.ONCLICK | Event.ONBLUR);
        hasWords = new HashMap<String, HasWord>();
        generateHasWords();
    }

    private void generateHasWords() {
        for (ChronoBands band : ChronoBands.values()) {
            for (int x = 0; x < 5; x++) {
```

```
                String elementID = band.getIDPrefix() + "_" + (x + 1);
                final Label l = new Label();
                mainArea.addAndReplaceElement(l, elementID);
                l.setStylePrimaryName(TREND_WORD);
                HasWord hw = new HasWord() {
                    @Override
                    public HasText getText() {
                        return l;
                    }

                    @Override
                    public HasClickHandlers getClickHandlers() {
                        return l;
                    }
                };
                hasWords.put(elementID, hw);
            }
        }
    }

    @Override
    public HasWord getWord(ChronoBands band, int position) {
        String elementID = band.getIDPrefix() + "_" + (position + 1);
        return hasWords.get(elementID);
    }


    private static final String TRENDS_LIST =
" <ul class=\"trends\"> \n"+
" <li class=\"chrono_label\">this hour:</li> \n"+
" <li class=\"trend_word\"><a id=\"h_1\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"h_2\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"h_3\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"h_4\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"h_5\"></a></li>\n"+
" <li class=\"chrono_break\"></li>\n"+
" <li class=\"chrono_label\">today:</li>\n"+
" <li class=\"trend_word\"><a id=\"d_1\">...</a></li>\n"+
" <li class=\"trend_word\"><a id=\"d_2\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"d_3\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"d_4\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"d_5\"></a></li>\n"+
" <li class=\"chrono_break\"></li>\n"+
" <li class=\"chrono_label\">this week:</li>\n"+
" <li class=\"trend_word\"><a id=\"w_1\">...</a></li>\n"+
" <li class=\"trend_word\"><a id=\"w_2\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"w_3\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"w_4\"></a></li>\n"+
" <li class=\"trend_word\"><a id=\"w_5\"></a></li>\n"+
" </ul>";
}
```

Listing D.45: src/org/recursed/w/client/summary/SummaryPresenter.java

```
package org.recursed.w.client.summary;
```

```
import org.recursed.w.client.characteristics.HasWord;
import org.recursed.w.client.place.Place;
import org.recursed.w.client.place.PlaceChangeEvent;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.SummaryPlace;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;
import org.recursed.w.client.relationshipgraph.RelationshipGraphPresenter;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.GetTopWordsInChronoBands;
import org.recursed.w.shared.TopWordsInChronoBands;
import org.recursed.w.shared.WordDTO;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Timer;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.inject.Inject;

public class SummaryPresenter implements PlaceChangeHandler {

    private static final int PRESENT_WORDS_FOR = 1000 * 10;
    public interface Display {
        public HasWord getWord(ChronoBands band, int position);
    }

    private Display display;
    private RPCService rpcService;
    private PlaceService placeService;
    private TopWordsInChronoBands currentTopWords;
    private int currentMarker;
    private Timer timedPresenter;
    private static boolean justBooted = true;
    private boolean pickAWord = false;

    @Inject
    private SummaryPresenter(Display display, RPCService rpcService,
            PlaceService placeService,
            RelationshipGraphPresenter relationshipGraphPresenter) {
        placeService.addPlaceChangeHandler(this);
        this.display = display;
        this.placeService = placeService;
        this.rpcService = rpcService;
        Timer t = new Timer() {

            @Override
            public void run() {
                getData();
            }

        };
        // get data every 10 minutes
        t.scheduleRepeating(1000 * 60 * 10);
        getData();
```

```
        for (final ChronoBands band : ChronoBands.values()) {
            for (int x = 0; x < 5; x++) {
                HasWord hw = display.getWord(band, x);
                final int position = x;
                hw.getClickHandlers().addClickHandler(new ClickHandler() {
                    @Override
                    public void onClick(ClickEvent event) {
                        firePlaceChangeFor(band, position+(currentMarker-5));
                    }
                });
            }
        }
        timedPresenter = new Timer() {
            @Override
            public void run() {
                populate();
            }
        };
    }

    private void getData() {
        GetTopWordsInChronoBands get = new GetTopWordsInChronoBands();
        rpcService.execute(get, new GetTopWordsInChronoBandsCallback() {

            @Override
            public void got(TopWordsInChronoBands result) {
                currentTopWords = result;
                currentMarker = 0;
                resetTimerPresenter();
            }

            @Override
            public void onFailure(Throwable caught) {
                RootPanel.get().add(new Label("rpc failed"));
            }
        });
    }

    private void resetTimerPresenter() {
        populate();
        timedPresenter.scheduleRepeating(PRESENT_WORDS_FOR);
    }

    private void populate() {
        ChronoBands[] bands = ChronoBands.values();
        for (final ChronoBands band : bands) {
            final WordDTO[] words = currentTopWords.getWords(band);
            for (int x = 0; x<5 && x<words.length; x++) {
                HasWord hw = display.getWord(band, x);
                hw.getText().setText(words[currentMarker+x].getDisplayString());
                if (pickAWord) {
                    placeService.changePlace(new WordDetailPlace(words[currentMarker+x]));
                    pickAWord = false;
                }
            }
        }
        currentMarker += 5;
```

```
        if (currentMarker == 30) {
            currentMarker = 0;
        }
    }

    private void firePlaceChangeFor(ChronoBands band, int position) {
        WordDTO dto = currentTopWords.getWords(band)[position];
        WordDetailPlace w = new WordDetailPlace(dto);
        placeService.changePlace(w);
    }

    @Override
    public void onPlaceChangeEvent(PlaceChangeEvent pce) {
        Place p = pce.getPlace();
        if (p instanceof SummaryPlace) {
            if (justBooted) {
                pickAWord = true;
            }
        } else {
            // display.getVisibility().setVisibility(false);
        }
        justBooted = false;
    }

}
```

Listing D.46: src/org/recursed/w/client/W.java

```
package org.recursed.w.client;

import org.recursed.w.client.guice.WGinjector;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.core.client.GWT;

public class W implements EntryPoint {

    private final WGinjector injector = GWT.create(WGinjector.class);


    public void onModuleLoad() {
        injector.getAboutPresenter();
        injector.getSummaryPresenter();
        injector.getChosenWordPresenter();
        injector.getHeadlinesPresenter();
        injector.getPlaceService().fireInitialPlace();
        injector.getSearchPresenter();
    }
}
```

Listing D.47: src/org/recursed/w/client/worddetail/ChosenWordDisplay.java

```
package org.recursed.w.client.worddetail;

import java.util.ArrayList;

import org.recursed.w.client.characteristics.HasWord;
import org.recursed.w.client.worddetail.ChosenWordPresenter.HasTrendArea;

import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlowPanel;
import com.google.gwt.user.client.ui.HasText;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class ChosenWordDisplay implements ChosenWordPresenter.Display {

    private static final String DIV_CHOSEN_WORD = "chosen_word_container";
    private static final String DATES = "t_dates";
    private static final String WORD = "t_word";
    private static final String T_WORD_DIV = "t_words";

    private ArrayList<TrendArea> trendAreas;
    private int counter;

    @Inject
    public ChosenWordDisplay() {
        RootPanel rootPanel = RootPanel.get(DIV_CHOSEN_WORD);
        trendAreas = new ArrayList<TrendArea>();
        counter =0;
        for (int x = 0; x<10; x++) {
            TrendArea t = new TrendArea();
            rootPanel.add(t);
            trendAreas.add(t);
        }
    }

    @Override
    public HasTrendArea getTrendArea() {
        TrendArea trendArea = trendAreas.get(counter);
        counter++;
        return trendArea;
    }

    private class TrendArea extends Composite implements HasTrendArea {

        private FlowPanel div;
        private Label dates;
        private FlowPanel words;

        TrendArea () {
            div = new FlowPanel();
            dates = new Label();
            dates.setStyleName(DATES);
            div.add(dates);
```

```
            words = new FlowPanel();
            words.setStyleName(T_WORD_DIV);
            div.add(words);
            initWidget(div);
        }

        @Override
        public HasWord getNextWord() {
            final Label l = new Label();
            l.setStyleName(WORD);
            words.add(l);
            HasWord result = new HasWord() {

                @Override
                public HasClickHandlers getClickHandlers() {
                    return l;
                }

                @Override
                public HasText getText() {
                    return l;
                }

            };
            return result;
        }

        public void reset() {
            dates.setText(null);
            words.clear();
        }

        @Override
        public void setDateRange(String string) {
            dates.setText(string);
        }
    }

    @Override
    public void reset() {
        counter = 0;
        for (TrendArea ta : trendAreas) {
            ta.reset();
        }
    }

}
```

Listing D.48: src/org/recursed/w/client/worddetail/ChosenWordPresenter.java

```
package org.recursed.w.client.worddetail;

import java.util.ArrayList;
```

```
import org.recursed.w.client.characteristics.HasWord;
import org.recursed.w.client.place.PlaceChangeEvent;
import org.recursed.w.client.place.PlaceService;
import org.recursed.w.client.place.WordDetailPlace;
import org.recursed.w.client.place.PlaceChangeEvent.PlaceChangeHandler;
import org.recursed.w.client.rpc.RPCService;
import org.recursed.w.shared.GetWordDetailAction;
import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.WordDetail;
import org.recursed.w.shared.trends.TrendDTO;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.event.dom.client.HasClickHandlers;
import com.google.gwt.event.shared.HandlerRegistration;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class ChosenWordPresenter implements PlaceChangeHandler {

    public interface Display {

        public void reset();
        public HasTrendArea getTrendArea();
    }


    public interface HasTrendArea {

        public HasWord getNextWord();
        public void setDateRange(String string);
    }

    private Display display;
    private RPCService rpcService;
    private PlaceService placeService;
    private ArrayList<HandlerRegistration> cleanup = new ArrayList<HandlerRegistration>();
    private int currentlyFetching;

    @Inject
    private ChosenWordPresenter(PlaceService placeService,
            RPCService rpcService, Display display) {
        placeService.addPlaceChangeHandler(this);
        this.rpcService = rpcService;
        this.display = display;
        this.placeService = placeService;
    }

    private void cleanupRegistration() {
        for (HandlerRegistration h : cleanup) {
            h.removeHandler();
        }
        cleanup = new ArrayList<HandlerRegistration>();
    }
```

```
    private void populateWord(int id) {
        this.currentlyFetching = id;
        GetWordDetailAction gwa = new GetWordDetailAction(id, GetWordDetailAction.
            DEFAULT_MAX_WORDS);
        rpcService.execute(gwa, new GetWordDetailCallback() {

            @Override
            public void got(WordDetail response) {
                WordDTO coreWord = response.getCoreWord();
                if (coreWord.getId() != currentlyFetching) {
                    return;
                }
                display.reset();
                ArrayList<TrendDTO> trends = response.getTrends();
                for (TrendDTO trend : trends) {
                    ArrayList<String> dates = trend.getDates();
                    ArrayList<WordDTO> words = trend.getWords();
                    String startDate = dates.get(0);
                    String endDate = dates.get(dates.size()-1);
                    HasTrendArea trendArea = display.getTrendArea();
                    trendArea.setDateRange("From " + startDate + " to " + endDate + ":");
                    for (WordDTO word : words) {
                        HasWord hasWord = trendArea.getNextWord();
                        hasWord.getText().setText(word.getDisplayString());
                        registerForClicks(hasWord, word);
                    }
                }
            }
        });
    }

    private void registerForClicks(HasWord hasWord, final WordDTO word) {
        HasClickHandlers clickHandlers = hasWord.getClickHandlers();
        HandlerRegistration addClickHandler = clickHandlers.addClickHandler(new ClickHandler
            () {

            @Override
            public void onClick(ClickEvent event) {
                placeService.changePlace(new WordDetailPlace(word));
            }
        });
        cleanup.add(addClickHandler);
    }


    @Override
    public void onPlaceChangeEvent(PlaceChangeEvent pce) {
        if (pce.getPlace() instanceof WordDetailPlace) {
            cleanupRegistration();
            WordDetailPlace wdp = (WordDetailPlace) pce.getPlace();
            int id = wdp.getWordId();
            populateWord(id);
        }
    }


}
```

Listing D.49: src/org/recursed/w/client/worddetail/GetWordDetailCallback.java

```
package org.recursed.w.client.worddetail;

import org.recursed.w.client.rpc.ActionCallback;
import org.recursed.w.shared.WordDetail;

public abstract class GetWordDetailCallback extends ActionCallback<WordDetail>{

}
```

Listing D.50: src/org/recursed/w/server/ActionHandler.java

```
package org.recursed.w.server;

import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.Response;
/**
 * An interface that provides an RPC contract.
 * @author keerat
 * @param <A> the inbound Action to be serviced
 * @param <R> the type of Response to be packaged as a return
 */
public interface ActionHandler <A extends Action, R extends Response> {

    /**
     * Handle the processing of the supplied Action
     * @param action
     * @return a Response value
     */
    public R execute(A action);

}
```

Listing D.51: src/org/recursed/w/server/handlers/BatchedActionHandler.java

```
package org.recursed.w.server.handlers;

import java.util.HashMap;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.BatchedAction;
import org.recursed.w.shared.corerpc.BatchedResponse;
import org.recursed.w.shared.corerpc.Response;

public class BatchedActionHandler implements ActionHandler<BatchedAction, BatchedResponse>{

    private HandlerRegistry registry;
    //private ExecutorService threadpool = Executors.newCachedThreadPool();
```

```
    @Override
    public BatchedResponse execute(BatchedAction action) {
        HashMap<Integer,Action> actions = action.getActions();
        final BatchedResponse batchedResponse = new BatchedResponse();
        //StringBuilder batchData = new StringBuilder("Batch contains:");
        for (Integer key : actions.keySet()) {
            Action a = actions.get(key);
            //batchData.append("\n ").append(key).append(':').append(a.getClass());
            try {
                Response response = registry.execute(a);
                batchedResponse.addResponse(key, response);
            } catch (Exception e) {
                batchedResponse.addException(key, e);
            }
        }
        //System.out.println(batchData);
        return batchedResponse;
    }

    public void setRegistry(HandlerRegistry handlerRegistry) {
        this.registry = handlerRegistry;
    }

}
```

Listing D.52: src/org/recursed/w/server/handlers/GetHeadlinesHandler.java

```
package org.recursed.w.server.handlers;

import java.util.ArrayList;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.sql.WordsDAO;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.headlines.GetHeadlines;
import org.recursed.w.shared.headlines.GetHeadlinesResponse;
import org.recursed.w.shared.headlines.HeadlineDTO;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class GetHeadlinesHandler implements ActionHandler<GetHeadlines, GetHeadlinesResponse
    >{

    private static final long DURATION = 4l* ChronoBands.WEEK.getRangeInMillis();

    private WordsDAO wordsDAO;

    @Inject
    public GetHeadlinesHandler(WordsDAO wordsDAO) {
        this.wordsDAO = wordsDAO;
    }
```

```
    @Override
    public GetHeadlinesResponse execute(GetHeadlines action) {
        GetHeadlinesResponse response = new GetHeadlinesResponse();
        ArrayList<HeadlineDTO> headlines = wordsDAO.getHeadlines(action.getWordId(), System.
            currentTimeMillis() - DURATION);
        response.addHeadlines(headlines);
        return response;
    }

}
```

Listing D.53: src/org/recursed/w/server/handlers/GroupedWordDetailHandler.java

```
package org.recursed.w.server.handlers;

import java.util.ArrayList;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.sql.RelatedWordResult;
import org.recursed.w.server.sql.WordsDAO;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.GetGroupedWordDetailAction;
import org.recursed.w.shared.GroupedWordDetail;
import org.recursed.w.shared.WordDTO;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class GroupedWordDetailHandler implements
        ActionHandler<GetGroupedWordDetailAction, GroupedWordDetail> {

    private WordsDAO dao;

    @Inject
    public GroupedWordDetailHandler(WordsDAO dao) {
        this.dao = dao;
    }

    @Override
    public GroupedWordDetail execute(GetGroupedWordDetailAction action) {
        ArrayList<WordDTO> sourceWords = action.getWords();
        GroupedWordDetail result = new GroupedWordDetail();
        long now = System.currentTimeMillis();
        long start = now - (ChronoBands.WEEK.getRangeInMillis() * 41);
        for (WordDTO source : sourceWords) {
            RelatedWordResult rwr = dao.getRelatedWords(source.getId(), start);
            ArrayList<WordDTO> computed = rwr.compute(5);
            for (int x=0; x<computed.size(); x++) {
                if (computed.get(x) != null) {
                    result.addLinkage(source, computed.get(x));
                } else {
                    break;
```

```
                }
            }
        }
        return result;
    }

}
```

Listing D.54: src/org/recursed/w/server/handlers/HandlerRegistry.java

```
package org.recursed.w.server.handlers;

import java.util.HashMap;
import java.util.Map;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.servlet.Histogram;
import org.recursed.w.shared.GetGroupedWordDetailAction;
import org.recursed.w.shared.GetTopWordsInChronoBands;
import org.recursed.w.shared.GetWordDetailAction;
import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.BatchedAction;
import org.recursed.w.shared.corerpc.Response;
import org.recursed.w.shared.headlines.GetHeadlines;
import org.recursed.w.shared.search.GetSuggestedWordsAction;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class HandlerRegistry {

    private Map<Class<? extends Action>, ActionHandler<? extends Action, ? extends Response
        >> handlers;
    private Histogram histogram;

    @Inject
    public HandlerRegistry(Histogram histogram,
            TopWordsInChronoBandsHandler twicbh, WordDetailHandler wdh,
            GroupedWordDetailHandler gwdh, BatchedActionHandler bah, GetHeadlinesHandler ghh,
            SuggestWordHandler swh) {
        this.handlers = new HashMap<Class<? extends Action>, ActionHandler<? extends Action
            , ? extends Response>>();
        handlers.put(GetTopWordsInChronoBands.class, twicbh);
        handlers.put(GetWordDetailAction.class, wdh);
        handlers.put(GetGroupedWordDetailAction.class, gwdh);
        handlers.put(BatchedAction.class, bah);
        handlers.put(GetHeadlines.class, ghh);
        handlers.put(GetSuggestedWordsAction.class, swh);
        bah.setRegistry(this);
        this.histogram = histogram;
    }

    @SuppressWarnings("unchecked")
    public <R extends Response> R execute(Action action) {
```

```
        ActionHandler handler = getActionHandler(action);
        R r = null;
        long start = System.currentTimeMillis();
        try {
            r = (R) handler.execute(action);
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        } finally {
            long end = System.currentTimeMillis();
            histogram.record(handler, start, end);
        }
        return r;
    }

    /**
     *
     * @param <A>
     * @param a
     * @return
     */
    @SuppressWarnings("unchecked")
    public <A extends Action> ActionHandler<A, ? extends Response> getActionHandler(
            A a) {
        return (ActionHandler<A, ?>) handlers.get(a.getClass());
    }
}
```

Listing D.55: src/org/recursed/w/server/handlers/SuggestWordHandler.java

```
package org.recursed.w.server.handlers;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.sql.WordsDAO;
import org.recursed.w.shared.search.GetSuggestedWordsAction;
import org.recursed.w.shared.search.SuggestedWordsResponse;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class SuggestWordHandler implements ActionHandler<GetSuggestedWordsAction,
     SuggestedWordsResponse>{

    private WordsDAO dao;

    @Inject
    public SuggestWordHandler(WordsDAO dao) {
        this.dao = dao;
    }

    @Override
    public SuggestedWordsResponse execute(GetSuggestedWordsAction action) {
        String segment = action.getSegment();
        return dao.getSuggestions(segment);
```

```
    }
}
```

Listing D.56: src/org/recursed/w/server/handlers/TopWordsInChronoBandsHandler.java

```
package org.recursed.w.server.handlers;

import java.util.List;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.sql.WordsDAO;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.GetTopWordsInChronoBands;
import org.recursed.w.shared.TopWordsInChronoBands;
import org.recursed.w.shared.WordDTO;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class TopWordsInChronoBandsHandler implements
        ActionHandler<GetTopWordsInChronoBands, TopWordsInChronoBands> {

    private WordsDAO wordsDAO;
    private TopWordsInChronoBands staleResult;
    private long lastAccess = 0;
    private static final long FIVE_MINS = 1000l*60l*5l;

    @Inject
    private TopWordsInChronoBandsHandler(WordsDAO wordsDAO) {
        this.wordsDAO = wordsDAO;
    }

    @Override
    public TopWordsInChronoBands execute(GetTopWordsInChronoBands action) {
        TopWordsInChronoBands result = checkCache(action);
        if (result == null) {
            result = getFromDatabase(action);
        }
        return result;
    }

    private TopWordsInChronoBands checkCache(final GetTopWordsInChronoBands action) {
        TopWordsInChronoBands result = staleResult;
        if (lastAccess < System.currentTimeMillis()- FIVE_MINS) {
            lastAccess = System.currentTimeMillis();
            System.out.println("STALE_CACHE in TopWordsInChronoBandsHandler");
            Runnable r = new Runnable() {
                @Override
                public void run() {
                    getFromDatabase(action);
                }
```

```
            };
            new Thread(r).start();
        }
        return result;
    }

    private TopWordsInChronoBands getFromDatabase(GetTopWordsInChronoBands action) {
        TopWordsInChronoBands result = new TopWordsInChronoBands();
        ChronoBands[] bands = ChronoBands.values();
        long finish = System.currentTimeMillis();
        int n = action.getNumberOfWords();
        for (ChronoBands band : bands) {
            long start = finish - band.getRangeInMillis();
            List<WordDTO> words = wordsDAO.getWords(n, start, finish);
            WordDTO[] wordArr = new WordDTO[words.size()];
            result.setWords(band, words.toArray(wordArr));
        }
        staleResult = result;
        return result;
    }
}
```

Listing D.57: src/org/recursed/w/server/handlers/WordDetailHandler.java

```
package org.recursed.w.server.handlers;

import java.util.ArrayList;

import org.recursed.w.server.ActionHandler;
import org.recursed.w.server.sql.RelatedWordResult;
import org.recursed.w.server.sql.TrendCalculator;
import org.recursed.w.server.sql.WordsDAO;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.GetWordDetailAction;
import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.WordDetail;
import org.recursed.w.shared.trends.TrendDTO;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class WordDetailHandler implements
        ActionHandler<GetWordDetailAction, WordDetail> {

    private WordsDAO dao;

    @Inject
    public WordDetailHandler(WordsDAO dao) {
        this.dao = dao;
    }

    @Override
    public WordDetail execute(GetWordDetailAction action) {
        int id = action.getWordId();
```

```
        WordDTO coreWord = dao.getWord(id);
        long now = System.currentTimeMillis();
        WordDetail response = new WordDetail(coreWord);
        ChronoBands band = ChronoBands.WEEK;
        long diff = 4l*(band.getRangeInMillis());
        long start = now - diff;
        RelatedWordResult rwr = dao.getRelatedWords(id, start);
        TrendCalculator tc = new TrendCalculator(rwr);
        ArrayList<TrendDTO> trends = tc.getTrends(action.getMaxWords());
        response.setTrends(trends);
        return response;
    }

}
```

Listing D.58: src/org/recursed/w/server/servlet/ActionHandlerServlet.java

```
package org.recursed.w.server.servlet;

import javax.servlet.http.HttpServletRequest;

import org.recursed.w.server.handlers.HandlerRegistry;
import org.recursed.w.shared.corerpc.Action;
import org.recursed.w.shared.corerpc.ActionService;
import org.recursed.w.shared.corerpc.Response;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;
import com.google.gwt.user.server.rpc.SerializationPolicy;
import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
@SuppressWarnings("serial")
public class ActionHandlerServlet extends RemoteServiceServlet implements
        ActionService {

    private HandlerRegistry registry;

    @Inject
    private ActionHandlerServlet(HandlerRegistry registry) {
        this.registry = registry;
    }

    @SuppressWarnings("unchecked")
    @Override
    public <R extends Response> R execute(Action action) {
        return (R) registry.execute(action);
    }


    @Override
    protected SerializationPolicy doGetSerializationPolicy(
            HttpServletRequest request, String moduleBaseURL, String strongName) {
        //get the base url from the header instead of the body this way
        //apache reverse proxy with rewrite on the header can work
```

```
        String moduleBaseURLHdr = request.getHeader("X-GWT-Module-Base");
        //System.out.println("Resolved header to: " + moduleBaseURLHdr);
        if(moduleBaseURLHdr != null){
            moduleBaseURL = moduleBaseURLHdr;
        }
        if (moduleBaseURL.indexOf("/w/w") < 0) {
            moduleBaseURL = moduleBaseURL+"w/";
        }
        //System.out.println("Wrapped up with: " + moduleBaseURL + " " + strongName);
        return super.doGetSerializationPolicy(request, moduleBaseURL, strongName);
    }

}
```

Listing D.59: src/org/recursed/w/server/servlet/GuiceServletConfig.java

```
package org.recursed.w.server.servlet;

import com.google.inject.Guice;
import com.google.inject.Injector;
import com.google.inject.Module;
import com.google.inject.servlet.GuiceServletContextListener;

public class GuiceServletConfig extends GuiceServletContextListener {

    @Override
    protected Injector getInjector() {
        return Guice.createInjector(new Module[]{new GuiceServletConfigModule()});
    }

}
```

Listing D.60: src/org/recursed/w/server/servlet/GuiceServletConfigModule.java

```
package org.recursed.w.server.servlet;

import com.google.inject.servlet.ServletModule;

public class GuiceServletConfigModule extends ServletModule {

    @Override
    protected void configureServlets() {
        serve("*/arpc").with(ActionHandlerServlet.class);
        serve("*/stats").with(StatsServlet.class);
    }

}
```

Listing D.61: src/org/recursed/w/server/servlet/Histogram.java

```
package org.recursed.w.server.servlet;

import java.util.HashMap;
import java.util.Set;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public final class Histogram {

    private HashMap<Class<?>, HashMap<Integer, Integer>> data;

    @Inject
    protected Histogram() {
        data = new HashMap<Class<?>, HashMap<Integer, Integer>>();
    }

    public void record(Object o, long start, long finish) {
        try {
            Long longDuration = finish - start;
            int duration = longDuration.intValue()/1000;
            HashMap<Integer, Integer> histogram = data.get(o.getClass());
            if (histogram == null) {
                histogram = new HashMap<Integer, Integer>();
            }
            Integer tally = histogram.get(duration);
            if (tally == null) {
                tally = 0;
            }
            tally++;
            histogram.put(duration, tally);
            data.put(o.getClass(), histogram);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public Set<Class<?>> getAllKeys() {
        return data.keySet();
    }

    public HashMap<Integer, Integer> getData(Class<?> c) {
        return data.get(c);
    }

}
```

Listing D.62: src/org/recursed/w/server/servlet/StatsServlet.java

```
package org.recursed.w.server.servlet;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.HashMap;
import java.util.Set;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.recursed.w.server.sql.Pool;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
@SuppressWarnings("serial")
public class StatsServlet extends HttpServlet {

    private Runtime runtime;
    private Pool pool;
    private Histogram histogram;

    @Inject
    public StatsServlet(Pool pool, Histogram h) {
        this.pool = pool;
        this.histogram = h;
    }

    @Override
    protected void service(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/plain");
        StringBuilder sb = new StringBuilder();
        appendServerStats(sb);
        appendDBStats(sb);
        appendHistogram(sb);
        response.getWriter().print(sb.toString());
    }

    private void appendServerStats(StringBuilder sb) {
        if (runtime == null) {
            synchronized (this) {
                runtime = Runtime.getRuntime();
            }
        }
        Date d = new Date();
        sb.append(d).append('\n').append(runtime.freeMemory()).append('/').append(runtime.
             totalMemory()).append('\n');
    }


    private void appendDBStats(StringBuilder sb) {
        Connection con = pool.getConnection();
        PreparedStatement st = null;
        ResultSet rs = null;
```

```
        try {
            st = con.prepareStatement("select count(*) from words");
            rs = st.executeQuery();
            if (rs.next()) {
                sb.append("Word count: ").append(rs.getInt(1)).append("\n");
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        } finally {
            Pool.close(con, st, rs);
        }
    }

    private void appendHistogram(StringBuilder sb) {
        Set<Class<?>> classes = histogram.getAllKeys();
        sb.append("Histogram Data:\n");
        for (Class<?> c : classes) {
            sb.append(" ").append(c.getName()).append("\n");
            HashMap<Integer, Integer> data = histogram.getData(c);
            for (Integer bucket : data.keySet()) {
                Integer tally = data.get(bucket);
                sb.append(" ").append(bucket).append(":").append(tally).append("\n");
            }
        }
    }


}
```

Listing D.63: src/org/recursed/w/server/sql/Pool.java

```
package org.recursed.w.server.sql;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.sql.DataSource;

import org.apache.commons.dbcp.BasicDataSource;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class Pool {

    private DataSource dataSource;

    @Inject
    public Pool() {
        BasicDataSource ds = new BasicDataSource();
        ds.setDriverClassName("com.mysql.jdbc.Driver");
        ds.setUsername("");
```

```
        ds.setPassword("");
        ds.setUrl("jdbc:mysql://localhost:3306/w");
        ds.setValidationQuery("select 1");
        ds.setTestOnBorrow(true);
        this.dataSource = ds;
    }

    public Connection getConnection() {
        try {
            return dataSource.getConnection();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    public static void close(ResultSet rs) {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void close(Statement s) {
        try {
            if (s != null) {
                s.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void close(Connection con) {
        try {
            if (con != null) {
                con.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void close(Connection con, Statement s) {
        close(s);
        close(con);
    }

    public static void close(Connection con, Statement s, ResultSet rs) {
        close(rs);
        close(s);
        close(con);
    }
}
```

Listing D.64: src/org/recursed/w/server/sql/RelatedWordResult.java

```
package org.recursed.w.server.sql;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.TreeSet;

import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.WordDTO;

public class RelatedWordResult {

    private static final SimpleDateFormat DATE_FORMAT = new SimpleDateFormat("yyyy-MM-dd");
    private HashMap<Integer, Integer> wordFreqSum = new HashMap<Integer, Integer>();
    private HashMap<Integer, HashSet<Integer>> wordsInTally= new HashMap<Integer, HashSet<
        Integer>>();
    private HashMap<Integer, ArrayList<Tuple>> wordData = new HashMap<Integer, ArrayList<
        Tuple>>();
    private HashMap<Integer, WordDTO> wordDTOs = new HashMap<Integer, WordDTO>();
    private Date start = new Date();

    public RelatedWordResult(Date start) {
        this.start = start;
    }

    public void addWord(WordDTO word, Date d) {
        int wordId = word.getId();
        Integer freqSumI = wordFreqSum.get(wordId);
        int freqSum = 0;
        if (freqSumI != null) {
            freqSum = freqSumI;
        }
        HashSet<Integer> currentTallyGroup = wordsInTally.get(freqSum);
        if (currentTallyGroup != null) {
            currentTallyGroup.remove(wordId);
        }
        freqSum += word.getCount();
        HashSet<Integer> newTallyGroup = wordsInTally.get(freqSum);
        if (newTallyGroup == null) {
            newTallyGroup = new HashSet<Integer>();
        }
        newTallyGroup.add(wordId);
        wordsInTally.put(freqSum, newTallyGroup);
        wordFreqSum.put(wordId, freqSum);
        ArrayList<Tuple> currentWordData = wordData.get(wordId);
        if (currentWordData == null) {
            currentWordData = new ArrayList<Tuple>();
        }
        Tuple t = new Tuple(d, word.getCount());
        currentWordData.add(t);
        wordData.put(wordId, currentWordData);
        wordDTOs.put(wordId, word);
```

```
}


private class Tuple {
    String date;
    int count;
    Tuple(Date d, int count) {
        this.date = DATE_FORMAT.format(d);
        this.count = count;
    }
}

public ArrayList<WordDTO> compute(int maxWords) {
    TreeSet<Integer> rankedTallys = new TreeSet<Integer>(wordsInTally.keySet());
    Iterator<Integer> it = rankedTallys.descendingIterator();
    ArrayList<WordDTO> result = new ArrayList<WordDTO>();
    int counter = 0;
    while (it.hasNext() && counter < maxWords) {
        Integer tally = it.next();
        HashSet<Integer> wordIDs = wordsInTally.get(tally);
        Iterator<Integer> wordIDIterator = wordIDs.iterator();
        while (wordIDIterator.hasNext() && counter < maxWords) {
            int wordId = wordIDIterator.next();
            WordDTO dto = wordDTOs.get(wordId);
            result.add(dto);
            counter++;
        }
    }
    return result;
}

public int[] getValues(WordDTO w) {
    int id = w.getId();
    ArrayList<Tuple> tuples = wordData.get(id);
    int[] result = new int[28];
    int pointer = 0;
    long curDate = start.getTime();
    for (int x=0; x<result.length; x++) {
        String curDateString = DATE_FORMAT.format(curDate);
        if (pointer == tuples.size()) {
            // no more data to read..
            break;
        }
        if (curDateString.equals(tuples.get(pointer).date)) {
            result[x] = tuples.get(pointer).count;
            pointer++;
        }
        curDate += ChronoBands.DAY.getRangeInMillis();
    }
    return result;
}

public String[] getDates() {
    String[] result = new String[28];
    long now = System.currentTimeMillis();
    long curDate = start.getTime();
    for (int x=0; now > curDate && x<28 ; x++) {
```

```
            result[x] = DATE_FORMAT.format(curDate);
            curDate += ChronoBands.DAY.getRangeInMillis();
        }
        return result;
    }

}
```

Listing D.65: src/org/recursed/w/server/sql/Test.java

```
package org.recursed.w.server.sql;

import java.util.ArrayList;

import org.recursed.w.server.handlers.WordDetailHandler;
import org.recursed.w.shared.ChronoBands;
import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.trends.TrendDTO;

import com.google.inject.Inject;


public class Test {

    @Inject
    public Test(WordDetailHandler handler, WordsDAO dao) {
        long now = System.currentTimeMillis();
        //RelatedWordResult rwr = dao.getRelatedWords(9961, now - ChronoBands.WEEK.
              getRangeInMillis()*4);
        RelatedWordResult rwr = dao.getRelatedWords(1976, now - ChronoBands.WEEK.
             getRangeInMillis()*4);
        TrendCalculator tc = new TrendCalculator(rwr);
        ArrayList<TrendDTO> trends = tc.getTrends(20);
        for (TrendDTO t : trends) {
            for (String s: t.getDates())
                System.out.print(s + " ");
            System.out.println();
            for (WordDTO w : t.getWords())
                System.out.println(w.getDisplayString());
            System.out.println();
        }
        System.out.println("Done");
    }

    public static void main(String[] args) {
        Pool pool = new Pool();
        WordsDAO wd = new WordsDAO(pool);
        WordDetailHandler h = new WordDetailHandler(wd);
        new Test(h, wd);
    }
}
```

Listing D.66: src/org/recursed/w/server/sql/TrendCalculator.java

```
package org.recursed.w.server.sql;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.TreeSet;

import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.trends.TrendDTO;

public class TrendCalculator {

    private RelatedWordResult relatedWordResult;
    private HashMap<Signature, ArrayList<WordDTO>> signatureToWordMap;
    private String[] dates;

    public TrendCalculator(RelatedWordResult relatedWordResult) {
        this.relatedWordResult = relatedWordResult;
        signatureToWordMap = new HashMap<Signature, ArrayList<WordDTO>>();
    }

    private void calculateTrends(int numberOfWords) {
        ArrayList<WordDTO> words = relatedWordResult.compute(numberOfWords);
        this.dates = relatedWordResult.getDates();
        for (WordDTO word : words) {
            int[] values = relatedWordResult.getValues(word);
            Signature sig = resolveSignature(values);
            //System.out.println(word.getDisplayString() + " sig: " + Arrays.toString(sig));
            ArrayList<WordDTO> wordsForSignature = signatureToWordMap.get(sig);
            if (wordsForSignature == null) {
                wordsForSignature = new ArrayList<WordDTO>();
            }
            wordsForSignature.add(word);
            sig.incrementWeight(values);
            signatureToWordMap.put(sig, wordsForSignature);
        }
    }

    public ArrayList<TrendDTO> getTrends(int maxNumber) {
        calculateTrends(20);
        TreeSet<Signature> sortedKeys = new TreeSet<Signature>(signatureToWordMap.keySet())
            ;
        ArrayList<TrendDTO> result = new ArrayList<TrendDTO>();
        int x= 0;
        for (Signature s: sortedKeys) {
            if (x == maxNumber) {
                break;
            }
            TrendDTO t = new TrendDTO();
            ArrayList<WordDTO> words = signatureToWordMap.get(s);
            t.addWords(words);
            ArrayList<String> ds = s.getMatchingDates();
            t.addDates(ds);
            result.add(t);
        }
        return result;
```

```
}

private Signature resolveSignature(int[] wordValuesInTime) {
    int[] result = new int[wordValuesInTime.length];
    for (int x=0; x<result.length; x++) {
        if (wordValuesInTime[x] > 0) {
            result[x] = 1;
        }
    }
    return new Signature(result);
}

class Signature implements Comparable<Signature> {
    int[] values;
    int weight;

    public Signature(int[] values) {
        this.values = values;
    }

    public void incrementWeight(int[] trueValues) {
        for (int v : trueValues) {
            if (v > 1) {
                v = 2;
            }
            weight += v;
        }
    }

    public ArrayList<String> getMatchingDates() {
        ArrayList<String> result = new ArrayList<String>();
        for (int x=0; x<values.length; x++) {
            if (values[x] != 0) {
                result.add(dates[x]);
            }
        }
        return result;
    }

    @Override
    public int hashCode() {
        long result = 0;
        long powTen = 1;
        for (int x = 0; x<values.length; x++) {
            result = result + (powTen * values[x]);
            powTen = powTen * 10;
        }
        return new Long(result).hashCode();
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Signature) {
            Signature other = (Signature) obj;
            if (this.hashCode() == other.hashCode()){
                return true;
            } else {
```

```
                return false;
            }
        } else {
            return false;
        }
    }

    @Override
    public int compareTo(Signature o) {
        return this.weight - o.weight;
    }

}
}
```

Listing D.67: src/org/recursed/w/server/sql/WordsDAO.java

```
package org.recursed.w.server.sql;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.headlines.HeadlineDTO;
import org.recursed.w.shared.search.SuggestedWordsResponse;

import com.google.inject.Inject;
import com.google.inject.Singleton;

@Singleton
public class WordsDAO {

    private static final String GET_WORDS_IN_RANGE = "select w.word_id, w.normal, sum(wh.frequency
        ) as count "
            + "from words w, articles a, word_histogram wh where "
            + "w.ignore = 0 and "
            + "a.analyzed_on > ? and "
            + "a.analyzed_on < ? and "
            + "wh.article_id = a.article_id and "
            + "wh.word_id = w.word_id "
            + "group by wh.word_id order by count desc " + "limit ?";

    private static final String GET_WORD = "select * from words w where w.word_id = ?";

    private static final String GET_RELATED_WORDS_TIMELINE = "select w.word_id, w.normal, sum(wh2
        .frequency) wcount, "
            + "date(a.analyzed_on) from words w, word_histogram wh, "
            + "word_histogram wh2, articles a "
            + "where wh.word_id = ? and "
```

```
        + "wh.article_id = a.article_id "
        + "and wh2.article_id = wh.article_id "
        + "and w.word_id = wh2.word_id "
        + "and w.ignore = 0 and wh2.word_id != wh.word_id "
        + "and a.analyzed_on > ? "
        + "group by w.word_id, 4 order by 4 asc ";

private static final String GET_HEADLINES_FOR_WORD = "select s.source_id, a.title, a.url from "
        + "sources s, articles a, word_histogram wh where "
        + "s.source_id = a.source_id and "
        + "a.article_id = wh.article_id and "
        + "a.analyzed_on > ? and wh.word_id = ? "
        + "order by a.analyzed_on";

private static final String GET_SUGGESTED_WORDS = "select w.word_id, w.normal "
        + "from words w where w.normal like ? "
        + "and w.ignore = 0 "
        + "order by w.normal limit 20";

private Pool pool;

@Inject
public WordsDAO(Pool pool) {
    this.pool = pool;
}

public List<WordDTO> getWords(int topN, long start, long finish) {
    Connection con = null;
    PreparedStatement st = null;
    ResultSet rs = null;
    Timestamp startDate = new Timestamp(start);
    Timestamp finishDate = new Timestamp(finish);
    ArrayList<WordDTO> result = new ArrayList<WordDTO>();
    long t1 = System.currentTimeMillis();
    try {
        con = pool.getConnection();
        st = con.prepareStatement(GET_WORDS_IN_RANGE);
        st.setTimestamp(1, startDate);
        st.setTimestamp(2, finishDate);
        st.setInt(3, topN);
        rs = st.executeQuery();
        while (rs.next()) {
            int wordId = rs.getInt(1);
            String label = rs.getString(2);
            int count = rs.getInt(3);
            WordDTO word = new WordDTO(wordId, label, count);
            result.add(word);
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    } finally {
        Pool.close(con, st, rs);
        long t2 = System.currentTimeMillis();
        System.out.println(generateLogLine("GET_WORDS_IN_RANGE", topN, start, finish, (
            t2-t1)));
    }
    return result;
```

```
}

public WordDTO getWord(int id) {
    Connection con = null;
    PreparedStatement st = null;
    ResultSet rs = null;
    WordDTO result;
    long t1 = System.currentTimeMillis();
    try {
        con = pool.getConnection();
        st = con.prepareStatement(GET_WORD);
        st.setInt(1, id);
        rs = st.executeQuery();
        if (rs.next()) {
            String normal = rs.getString("normal");
            result = new WordDTO(id, normal, 0);
        } else {
            throw new RuntimeException("Word not found with id: " + id);
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    } finally {
        Pool.close(con, st, rs);
        long t2 = System.currentTimeMillis();
        System.out.println(generateLogLine("GET_WORD", id, (t2-t1)));
    }
    return result;
}

public RelatedWordResult getRelatedWords(int id, long start) {
    Connection con = null;
    PreparedStatement st = null;
    ResultSet rs = null;
    Timestamp startDate = new Timestamp(start);
    RelatedWordResult result = new RelatedWordResult(new Date(start));
    long t1 = System.currentTimeMillis();
    try {
        con = pool.getConnection();
        st = con.prepareStatement(GET_RELATED_WORDS_TIMELINE);
        st.setInt(1, id);
        st.setTimestamp(2, startDate);
        rs = st.executeQuery();
        while (rs.next()) {
            int wordId = rs.getInt(1);
            String label = rs.getString(2);
            int count = rs.getInt(3);
            Date d = rs.getDate(4);
            WordDTO word = new WordDTO(wordId, label, count);
            result.addWord(word, d);
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    } finally {
        Pool.close(con, st, rs);
        long t2 = System.currentTimeMillis();
        System.out.println(generateLogLine("GET_RELATED_WORDS_TIMELINE", id, start,
            (t2-t1)));
```

```
        }
        return result;
    }

    public ArrayList<HeadlineDTO> getHeadlines(int wordId, long start) {
        ArrayList<HeadlineDTO> result = new ArrayList<HeadlineDTO>();
        Connection con = null;
        PreparedStatement st = null;
        ResultSet rs = null;
        Timestamp startDate = new Timestamp(start);
        long t1 = System.currentTimeMillis();
        try {
            con = pool.getConnection();
            st = con.prepareStatement(GET_HEADLINES_FOR_WORD);
            st.setTimestamp(1, startDate);
            st.setInt(2, wordId);
            rs = st.executeQuery();
            while (rs.next()) {
                int sourceId = rs.getInt(1);
                String title = rs.getString(2);
                String url = rs.getString(3);
                HeadlineDTO dto = new HeadlineDTO(title, url, sourceId);
                result.add(dto);
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        } finally {
            Pool.close(con, st, rs);
            long t2 = System.currentTimeMillis();
            System.out.println(generateLogLine("GET_HEADLINES_FOR_WORD", wordId, start, (
                t2-t1)));
        }
        return result;
    }

    public SuggestedWordsResponse getSuggestions(String segment) {
        Connection con = null;
        PreparedStatement st = null;
        ResultSet rs = null;
        SuggestedWordsResponse response = new SuggestedWordsResponse();
        long t1 = System.currentTimeMillis();
        try {
            con = pool.getConnection();
            st = con.prepareStatement(GET_SUGGESTED_WORDS);
            st.setString(1, segment + "%");
            rs = st.executeQuery();
            while (rs.next()) {
                int wordID = rs.getInt(1);
                String normal = rs.getString(2);
                WordDTO dto = new WordDTO(wordID, normal, -1);
                response.addSuggestion(dto);
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        } finally {
            Pool.close(con, st, rs);
            long t2 = System.currentTimeMillis();
```

```
            System.out.println(generateLogLine("GET_SUGGESTED_WORDS", segment, (t2-t1)));
        }
        return response;
    }


    private static String generateLogLine(Object ...objects) {
        StringBuilder sb = new StringBuilder("SQL");
        for (Object o : objects) {
            sb.append("|");
            if (o != null) {
                sb.append(o);
            } else {
                sb.append("null");
            }
        }
        return sb.toString();
    }
}
```

Listing D.68: src/org/recursed/w/shared/ChronoBands.java

```
package org.recursed.w.shared;

import java.io.Serializable;

import com.google.gwt.user.client.rpc.IsSerializable;

public enum ChronoBands implements IsSerializable, Serializable {
    HOUR(1, "h"), DAY(24, "d"), WEEK(24 * 7, "w");

    private ChronoBands() {
    }

    private int range;
    private String idPrefix;

    private ChronoBands(int range, String idPrefix) {
        this.range = range;
        this.idPrefix = idPrefix;
    }

    public int getRangeInHours() {
        return range;
    }

    public long getRangeInMillis() {
        return 1000L * 60L * 60L * new Long(range).longValue();
    }

    public String getIDPrefix() {
        return idPrefix;
    }
}
```

Listing D.69: src/org/recursed/w/shared/corerpc/Action.java

```
package org.recursed.w.shared.corerpc;

import java.io.Serializable;

import com.google.gwt.user.client.rpc.IsSerializable;

/**
 * A marker interface that identifies an RPC request.
 * @author keerat
 */
public interface Action extends IsSerializable, Serializable {

}
```

Listing D.70: src/org/recursed/w/shared/corerpc/ActionService.java

```
package org.recursed.w.shared.corerpc;


import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

/**
 * The general pipeline for all Actions and Responses.
 */
@RemoteServiceRelativePath("arpc")
public interface ActionService extends RemoteService {
   public <R extends Response> R execute(Action action);
}
```

Listing D.71: src/org/recursed/w/shared/corerpc/ActionServiceAsync.java

```
package org.recursed.w.shared.corerpc;


import com.google.gwt.user.client.rpc.AsyncCallback;

/**
 * The async counterpart of <code>ActionService</code>.
 */
public interface ActionServiceAsync {
    void execute(Action a, AsyncCallback<? extends Response> callback);
}
```

Listing D.72: src/org/recursed/w/shared/corerpc/BatchedAction.java

```
package org.recursed.w.shared.corerpc;
```

```
import java.util.HashMap;

public class BatchedAction implements Action {

    private HashMap<Integer, Action > actions = new HashMap<Integer, Action>();
    private transient HashMap<Action, Integer> invertedMap = new HashMap<Action, Integer>();

    private static final long serialVersionUID = 1L;

    public int addAction(int counter, Action a) {
        Integer result = invertedMap.get(a);
        if (result == null) {
            this.actions.put(counter, a);
            result = counter;
        }
        return result;
    }

    public HashMap<Integer, Action> getActions() {
        return actions;
    }
}
```

Listing D.73: src/org/recursed/w/shared/corerpc/BatchedResponse.java

```
package org.recursed.w.shared.corerpc;

import java.util.HashMap;

public class BatchedResponse implements Response{

    private static final long serialVersionUID = 1L;

    private HashMap<Integer, Response> responses = new HashMap<Integer, Response>();
    private HashMap<Integer, Exception> exceptions = new HashMap<Integer, Exception>();

    public void addResponse(Integer key, Response r) {
        responses.put(key, r);
    }

    public void addException(Integer key, Exception e) {
        exceptions.put(key, e);
    }

    public Response getResponse(Integer key) {
        return responses.get(key);
    }

    public Exception getException(Integer key) {
        return exceptions.get(key);
    }

}
```

Listing D.74: src/org/recursed/w/shared/corerpc/Response.java

```
package org.recursed.w.shared.corerpc;

import java.io.Serializable;

import com.google.gwt.user.client.rpc.IsSerializable;

/**
 * A marker interface that identifies a Response for a particular Action.
 * @author keerat
 */
public interface Response extends IsSerializable, Serializable {

}
```

Listing D.75: src/org/recursed/w/shared/GetGroupedWordDetailAction.java

```
package org.recursed.w.shared;

import java.util.ArrayList;

import org.recursed.w.shared.corerpc.Action;

public class GetGroupedWordDetailAction implements Action {

    private static final long serialVersionUID = 1L;

    private ArrayList<WordDTO> words;
    public GetGroupedWordDetailAction() {
        words = new ArrayList<WordDTO>();
    }

    public void addWord(WordDTO w) {
        words.add(w);
    }

    public ArrayList<WordDTO> getWords() {
        return words;
    }
}
```

Listing D.76: src/org/recursed/w/shared/GetTopWordsInChronoBands.java

```
package org.recursed.w.shared;

import org.recursed.w.shared.corerpc.Action;

public class GetTopWordsInChronoBands implements Action {

    private static final long serialVersionUID = 1L;
    private ChronoBands[] bands;
    private int n = 30;
```

```
    public GetTopWordsInChronoBands() {
        this.bands = ChronoBands.values();
    }

    public GetTopWordsInChronoBands(ChronoBands[] bands) {
        this.bands = bands;
    }

    public ChronoBands[] getChronoBands() {
        return bands;
    }

    public int getNumberOfWords() {
        return n;
    }

}
```

Listing D.77: src/org/recursed/w/shared/GetWordDetailAction.java

```
package org.recursed.w.shared;

import org.recursed.w.shared.corerpc.Action;

public class GetWordDetailAction implements Action {

    private static final long serialVersionUID = 1L;
    public static final int DEFAULT_MAX_WORDS = 20;

    @SuppressWarnings("unused")
    private GetWordDetailAction() {}

    private int id;
    private int maxWords;

    public GetWordDetailAction(int wordId, int maxWords) {
        this.id = wordId;
        this.maxWords = maxWords;
    }

    public int getWordId() {
        return id;
    }

    public int getMaxWords() {
        return maxWords;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        result = prime * result + maxWords;
```

```
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        GetWordDetailAction other = (GetWordDetailAction) obj;
        if (id != other.id)
            return false;
        if (maxWords != other.maxWords)
            return false;
        return true;
    }


}
```

Listing D.78: src/org/recursed/w/shared/GroupedWordDetail.java

```
package org.recursed.w.shared;

import java.util.ArrayList;
import java.util.HashMap;

import org.recursed.w.shared.corerpc.Response;

public class GroupedWordDetail implements Response {

    private static final long serialVersionUID = 1L;

    private HashMap<WordDTO, ArrayList<WordDTO>> relationships;

    public GroupedWordDetail() {
        relationships = new HashMap<WordDTO, ArrayList<WordDTO>>();
    }

    public void addLinkage(WordDTO source, WordDTO wordDTO) {
        ArrayList<WordDTO> relatedWords = relationships.get(source);
        if (relatedWords == null) {
            relatedWords = new ArrayList<WordDTO>();
        }
        relatedWords.add(wordDTO);
        relationships.put(source, relatedWords);
    }

    public HashMap<WordDTO, ArrayList<WordDTO>> getLinkages() {
        return relationships;
    }

}
```

Listing D.79: src/org/recursed/w/shared/headlines/GetHeadlines.java

```
package org.recursed.w.shared.headlines;

import org.recursed.w.shared.corerpc.Action;

public class GetHeadlines implements Action {

    private static final long serialVersionUID = 1L;
    private int wordId;

    @SuppressWarnings("unused")
    private GetHeadlines() {}

    public GetHeadlines(int wordId) {
        this.wordId = wordId;
    }

    public int getWordId() {
        return wordId;
    }
}
```

Listing D.80: src/org/recursed/w/shared/headlines/GetHeadlinesCallback.java

```
package org.recursed.w.shared.headlines;

import java.util.ArrayList;

import org.recursed.w.client.rpc.ActionCallback;

public abstract class GetHeadlinesCallback extends ActionCallback<GetHeadlinesResponse>{

    @Override
    public void got(GetHeadlinesResponse response) {
        gotHeadlines(response.getHeadlines());
    }

    public abstract void gotHeadlines(ArrayList<HeadlineDTO> headlines);
}
```

Listing D.81: src/org/recursed/w/shared/headlines/GetHeadlinesResponse.java

```
package org.recursed.w.shared.headlines;

import java.util.ArrayList;

import org.recursed.w.shared.corerpc.Response;

public class GetHeadlinesResponse implements Response {
```

```
    private static final long serialVersionUID = 1L;

    private ArrayList<HeadlineDTO> headlines;

    public GetHeadlinesResponse() {
        headlines = new ArrayList<HeadlineDTO>();
    }

    public void addHeadline(HeadlineDTO headline) {
        this.headlines.add(headline);
    }

    public ArrayList<HeadlineDTO> getHeadlines() {
        return headlines;
    }

    public void addHeadlines(ArrayList<HeadlineDTO> more) {
        headlines.addAll(more);
    }
}
```

Listing D.82: src/org/recursed/w/shared/headlines/HeadlineDTO.java

```
package org.recursed.w.shared.headlines;

import java.io.Serializable;

import com.google.gwt.user.client.rpc.IsSerializable;

public class HeadlineDTO implements Serializable, IsSerializable {

    private static final long serialVersionUID = 1L;

    private String title;
    private String link;
    private int sourceId;

    @SuppressWarnings("unused")
    private HeadlineDTO() {}

    public HeadlineDTO(String title, String link, int sourceId) {
        this.title = title;
        this.link = link;
        this.sourceId = sourceId;
    }

    public String getTitle() {
        if (title != null) {
            return title;
        } else {
            return link;
        }
    }
```

```
    public String getLink() {
        return link;
    }

    public int getSourceId() {
        return sourceId;
    }
}
```

Listing D.83: src/org/recursed/w/shared/search/GetSuggestedWordsAction.java

```
package org.recursed.w.shared.search;

import org.recursed.w.shared.corerpc.Action;

public class GetSuggestedWordsAction implements Action {

    private static final long serialVersionUID = 1L;
    private String segment;

    @SuppressWarnings("unused")
    private GetSuggestedWordsAction() {}

    public GetSuggestedWordsAction(String segment) {
        this.segment = segment;
    }

    public String getSegment() {
        return segment;
    }
}
```

Listing D.84: src/org/recursed/w/shared/search/SuggestedWordsResponse.java

```
package org.recursed.w.shared.search;

import java.util.ArrayList;

import org.recursed.w.shared.WordDTO;
import org.recursed.w.shared.corerpc.Response;

public class SuggestedWordsResponse implements Response {

    private static final long serialVersionUID = 1L;
    private ArrayList<WordDTO> words;

    public SuggestedWordsResponse() {
        words = new ArrayList<WordDTO>();
    }
```

```
    public void addSuggestion(WordDTO dto) {
        words.add(dto);
    }

    public ArrayList<WordDTO> getWords() {
        return words;
    }

}
```

Listing D.85: src/org/recursed/w/shared/TopWordsInChronoBands.java

```
package org.recursed.w.shared;

import java.util.HashMap;
import java.util.Map;

import org.recursed.w.shared.corerpc.Response;

public class TopWordsInChronoBands implements Response {

    private static final long serialVersionUID = 1L;
    private Map<ChronoBands, WordDTO[]> bands;

    public TopWordsInChronoBands() {
        this.bands = new HashMap<ChronoBands, WordDTO[]>();
    }

    public void setWords(ChronoBands band, WordDTO[] words) {
        bands.put(band, words);
    }
    public WordDTO[] getWords(ChronoBands band) {
        return bands.get(band);
    }
}
```

Listing D.86: src/org/recursed/w/shared/trends/TrendDTO.java

```
package org.recursed.w.shared.trends;

import java.io.Serializable;
import java.util.ArrayList;

import org.recursed.w.shared.WordDTO;

import com.google.gwt.user.client.rpc.IsSerializable;

public class TrendDTO implements Serializable, IsSerializable {

    private static final long serialVersionUID = 1L;
    private ArrayList<String> dates;
    private ArrayList<WordDTO> words;
```

```
    public TrendDTO() {
        this.dates = new ArrayList<String>();
        this.words = new ArrayList<WordDTO>();
    }

    public void addDates(ArrayList<String> dates) {
        this.dates.addAll(dates);
    }

    public void addWords(ArrayList<WordDTO> words) {
        this.words.addAll(words);
    }

    public ArrayList<String> getDates() {
        return dates;
    }

    public ArrayList<WordDTO> getWords() {
        return words;
    }
}
```

Listing D.87: src/org/recursed/w/shared/WordDetail.java

```
package org.recursed.w.shared;

import java.util.ArrayList;
import java.util.HashSet;

import org.recursed.w.shared.corerpc.Response;
import org.recursed.w.shared.trends.TrendDTO;

public class WordDetail implements Response {

    private static final long serialVersionUID = 1L;

    private WordDTO coreWord;
    private ArrayList<TrendDTO> trends;

    @SuppressWarnings("unused")
    private WordDetail() {
        trends = new ArrayList<TrendDTO>();
    }

    public WordDetail(WordDTO coreWord) {
        this.coreWord = coreWord;
    }

    public void setTrends(ArrayList<TrendDTO> trends) {
        this.trends = trends;
    }

    public ArrayList<TrendDTO> getTrends() {
        return trends;
```

```
    }

    public WordDTO getCoreWord() {
        return coreWord;
    }

    public HashSet<WordDTO> getRandomRelatedWords(int max) {
        HashSet<WordDTO> allWordDTOs = new HashSet<WordDTO>();
        for (TrendDTO t : trends) {
            ArrayList<WordDTO> words = t.getWords();
            allWordDTOs.addAll(words);
        }
        int counter = 0;
        HashSet<WordDTO> result = new HashSet<WordDTO>();
        for (WordDTO dto : allWordDTOs) {
            if (counter == 6) {
                break;
            }
            result.add(dto);
            counter++;
        }
        return result;
    }
}
```

Listing D.88: src/org/recursed/w/shared/WordDTO.java

```
package org.recursed.w.shared;

import java.io.Serializable;

import com.google.gwt.user.client.rpc.IsSerializable;
import com.google.gwt.user.client.ui.SuggestOracle.Suggestion;

/**
 * Represents a Word with both ID and presentation version.
 *
 * @author keerat
 */
public class WordDTO implements IsSerializable, Serializable, Suggestion {

    private static final long serialVersionUID = 1L;
    private int id, count;
    private String label;

    @SuppressWarnings("unused")
    private WordDTO() {
    }

    public WordDTO(int id, String label, int count) {
        this.id = id;
        this.label = label;
        this.count = count;
    }
```

```
    public int getId() {
        return id;
    }

    @Override
    public String getDisplayString() {
        return label;
    }

    public int getCount() {
        return count;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        WordDTO other = (WordDTO) obj;
        if (id != other.id)
            return false;
        return true;
    }

    @Override
    public String getReplacementString() {
        return label;
    }


}
```

D.4 Configuration

Listing D.89: src/get-ivy-via-ant.xml

```
<project xmlns:ivy="antlib:org.apache.ivy.ant">
 <property name="ivy.install.version" value="2.1.0-rc2" />
 <property name="ivy.jar.dir" value="${ivy.home}/lib" />
 <property name="ivy.jar.file" value="${ivy.jar.dir}/ivy.jar" />
 <target name="download-ivy" unless="offline">
  <mkdir dir="${ivy.jar.dir}" />
```

```
  <!--
   download Ivy from web site so that it can be used even without any
   special installation
  -->
  <get
   src="http://repo2.maven.org/maven2/org/apache/ivy/ivy/${ivy.install.version}/ivy-${ivy.install.
        version}.jar"
   dest="${ivy.jar.file}" usetimestamp="true" />
 </target>

 <target name="init-ivy" depends="download-ivy">
  <!--
   try to load ivy here from ivy home, in case the user has not already
   dropped it into ant's lib dir (note that the latter copy will always
   take precedence). We will not fail as long as local lib dir exists
   (it may be empty) and ivy is in at least one of ant's lib dir or the
   local lib dir.
  -->
  <path id="ivy.lib.path">
   <fileset dir="${ivy.jar.dir}" includes="*.jar" />
  </path>
  <taskdef resource="org/apache/ivy/ant/antlib.xml" uri="antlib:org.apache.ivy.ant"
   classpathref="ivy.lib.path" />
 </target>

 <target name="ivy-all" depends="init-ivy">
  <ivy:configure />
  <property name="w.ivy.module" value="${dir.src}/w-ivy-module.xml" />
  <ivy:resolve file="${w.ivy.module}"
    conf="compile, war" />
  <ivy:retrieve pattern="${w.lib.dir}/[conf]/[artifact]-[revision].[ext]" />
</target>


</project>
```

Listing D.90: src/org/recursed/w/jetty-w.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN" "http://jetty.
      mortbay.org/configure.dtd">
<configure class="org.mortbay.jetty.webapp.WebAppContext">
  <set name="contextPath">/w</set>
  <set name="war"><systemproperty name="jetty.home" default="."/>/webapps/w.war</set>
</configure>
```

Listing D.91: src/org/recursed/w/W.gwt.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='w'>
  <!-- Inherit the core Web Toolkit stuff. -->
  <inherits name='com.google.gwt.user.User'/>
```

```
  <inherits name="com.google.gwt.inject.Inject"/>

  <!-- Inherit the default GWT style sheet. You can change -->
  <!-- the theme of your GWT application by uncommenting -->
  <!-- any one of the following lines. -->
  <inherits name='com.google.gwt.user.theme.standard.Standard'/>
  <!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->
  <!-- <inherits name='com.google.gwt.user.theme.dark.Dark'/> -->
  <!-- Other module inherits -->
  <inherits name='com.google.gwt.widgetideas.GWTCanvas'/>


  <!-- Specify the app entry point class. -->
  <entry-point class='org.recursed.w.client.W'/>

  <!-- Specify the paths for translatable code -->
  <source path='client'/>
  <source path='shared'/>

</module>
```

Listing D.92: src/w-ivy-module.xml

```
<ivy-module version="2.0">
 <info organisation="org.recursed" module="w" />
 <configurations>
  <conf name="war" description="the jars needed in the war"/>
  <conf name="compile" extends="war" description="what's needed for compilation"/>
 </configurations>
 <dependencies>
  <dependency org="com.google.inject" name="guice" rev="2.0" conf="war->default"/>
  <dependency org="com.google.inject.extensions" name="guice-servlet" rev="2.0" conf="war->
      default" />
  <dependency org="com.googlecode.gwt.inject" name="gin" rev="1.0" conf="compile->default"/>
  <dependency org="com.google.gwt" name="gwt-servlet" rev="2.0.4" conf="war->default"/>
  <dependency org="com.google.gwt" name="gwt-user" rev="2.0.4" conf="compile->default"/>
  <dependency org="com.google.gwt" name="gwt-dev" rev="2.0.4" conf="compile->default"/>
  <!-- <dependency org="com.google.gwt" name="gwt-incubator" rev="2.0.1" conf="compile->
      default"/> -->
  <dependency org="mysql" name="mysql-connector-java" rev="5.1.13" conf="war->default"/>
  <dependency org="commons-dbcp" name="commons-dbcp" rev="1.4" conf="war->default"/>
 </dependencies>
</ivy-module>
```

Bibliography

[1] Hypertext Transfer Protocol

[2] The Atom Syndication Format

[3] RDF Site Summary (RSS) 1.0

[4] RSS 2.0 Specification, http://feed2.w3.org/docs/rss2.html

[5] RSS, From Wikipedia, http://en.wikipedia.org/wiki/Rss

[6] Java.net: Rome

[7] XML::FeedPP Parse/write/merge/edit RSS/RDF/Atom syndication feeds

[8] Papers on Link Grammar

Daniel Sleator and Davy Temperley. 1991. Parsing English with a Link Grammar. Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. Third International Workshop on Parsing Technologies.

John Lafferty, Daniel Sleator, and Davy Temperley. 1992. Grammatical Trigrams: A Probabilistic Model of Link Grammar. Proceedings of the AAAI Conference on Probabilistic Approaches to Natural Language, October, 1992.

Dennis Grinberg, John Lafferty and Daniel Sleator. 1995. A robust parsing algorithm for link grammars. Carnegie Mellon University Computer Science technical report CMU-CS-95-125, and Proceedings of the Fourth

International Workshop on Parsing Technologies, Prague, September, 1995.

[9] Lingua::LinkParser Perl module implementing the Link Grammar Parser by Sleator, Temperley and Lafferty at CMU

[10] Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14, no. 3, pp 130-137,

[11] The Porter Stemming Algorithm Source code for perl module.

[12] Database Management Systems (3rd Edition) Raghu Ramakrishnan, Johannes Gehrke

[13] The Business of Media: A Survival Guide (Kindle Single) Dignan, Larry

[14] The Jayson Blair Project; *How did he bamboozle the New York Times?* , http://www.slate.com/id/2082741/ Jack Shafer [8 May 2003]

[15] News Aggregation and Copyright Law on PlagiarismToday.com